(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

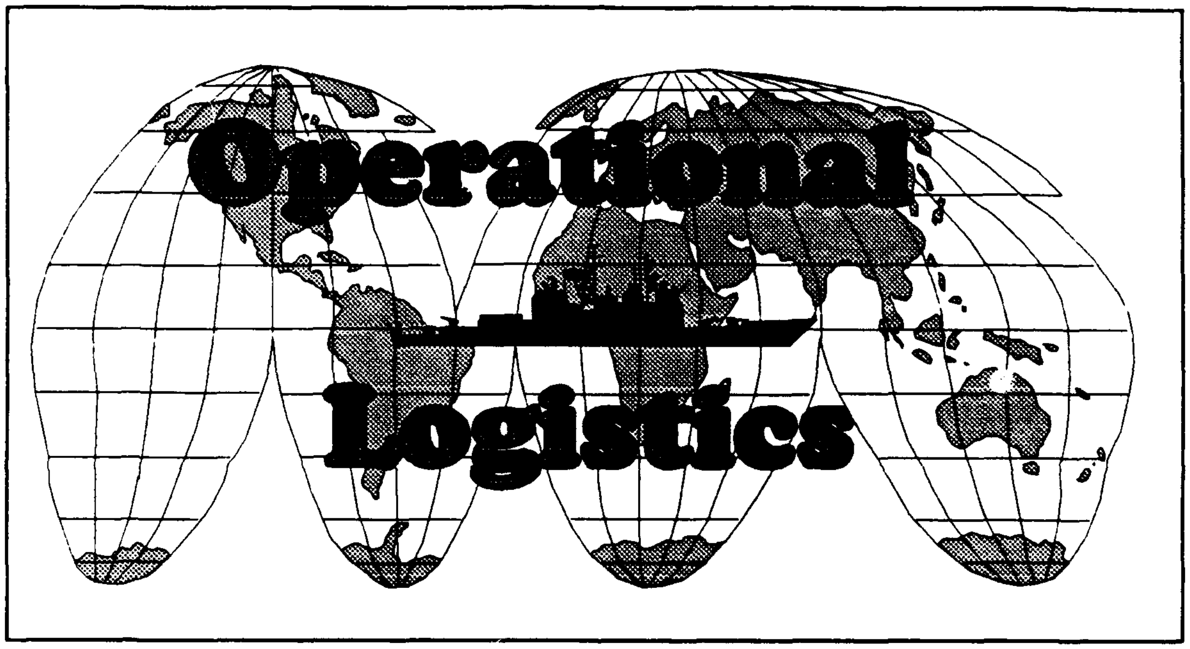
MODELING THEATER LEVEL LOGISTICS FOR WARGAMES

by

JOHN ARTHUR LONG

DECEMBER 1993

Thesis Advisor: David A. Schrady


Operational
Logistics

REPORT DOCUMENTATION PAGE	Form Approved OMB Np. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1993	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE MODELING THEATER LEVEL LOGISTICS FOR WARGAMES	5. FUNDING NUMBERS
6. AUTHOR(S) LONG, JOHN A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTA/RY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13.

ABSTRACT *(maximum 200 words)*

The Naval War College, Wargames Department needs a computer model that simulates theater level logistics to generate wargame "ground truth" and to aid players in simple planning. Of course, a logistics model must meet specific performance requirements in order to fill the needs of the Naval War College. This thesis presents the Surge and Sustainment Simulation, S3, a model with the required characteristics to allow the Naval War College to add logistical constraints to their wargames, both ENWGS and seminar. The relevant characteristics of the required model are defined, and the S3 model is described with a view to answering the stated requirements. Finally, an example of a Naval War College wargame run with the Surge and Sustainment Simulation is provided.

14. SUBJECT TERMS Logistics, Wargames, Deployment, Simulation	15. NUMBER OF 586
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)

/Prescribed by ANSI Std. 239-18

MODELING THEATER LEVEL LOGISTICS FOR WARGAMES

by

John Arthur Long
Lieutenant, United States Navy
B.S., Carnegie-Mellon University, 1986

Submitted in partial fulfillment
of the requirements for the degree

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
December 1993

Author: [illegible]
John Arthur Long

Approved by: [illegible]
David A. Schrady, Thesis Advisor

[illegible]
Thomas E. Halwachs, Second Reader

[illegible]
Peter Purdue, Chairman
Department of Operations Research

ABSTRACT

The Naval War College, Wargames Department needs a computer model that simulates theater level logistics to generate wargame "ground truth" and to aid players in simple planning. Of course, a logistics model must meet specific performance requirements in order to fill the needs of the Naval War College. This thesis presents the Surge and Sustainment Simulation, S3, a model with the required characteristics to allow the Naval War College to add logistical constraints to their wargames, both ENWGS and seminar. The relevant characteristics of the required model are defined, and the S3 model is described with a view to answering the stated requirements. Finally, an example of a Naval War College wargame run with the Surge and Sustainment Simulation is provided.

THESIS DISCLAIMER

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The Naval War College, Wargames Department needs a model to simulate theater level logistics as a constraint for wargame "ground truth" and to aid players in simple planning. This model must handle both surge and sustainment phases of the Major Regional Contingency. Model inputs should be based on nominal Logistics Planning factors and unit Tables of Organization and Equipment (TO&E). Preferably, the model should have, at the least, a menu-driven user interface with a built-in scenario generation routine to ease data entry. Model outputs must be, at a minimum, the logistics status of operating units and their supporting bases. If possible, the logistics status of the units and bases should be displayed in a simple, easily understood manner at different levels of aggregation. Additionally, the model should be flexible enough to handle unexpected game events while providing reasonable estimates of the logistics situation. This implies the ability to interface with the user and the wargame during simulation execution.

The Surge and Sustainment Simulation, S3 has been developed to meet this need. Written in the object-oriented MODSIM II simulation language, the model depicts theater logistics as a network with objects that represent nodes

(logistics sites, combat units), arcs (transportation assets), commodities, and national production.

Each node maintains an inventory of *Commodities* which are produced, shipped, and consumed. Each inventory item has, as defining characteristics, a stocking objective, an on-hand amount, an order point, and a count of items already on order. Commodities are moved from node to node via *Transporters* (ships, aircraft, trucks, and trains). Movement of Commodities from place to place is initiated by request from other nodes and expedited by a Logistics Manager object. Requests are driven by periodic checks of node inventory levels (once per simulated day). An overall logistics status is displayed to the user, automatically, at the command prompt. Additionally, the user may display any node, Transporter, or Commodity in detail through the menu system. Finally, the S3 allows the user to create and save objects to data files in order to build a database of transporters, units, and logistics nodes to ease scenario generation.

A demonstration of S3's capabilities is provided. The example scenario was developed by the Naval War College for use in the NOV 1993 Surface Warfare Officer School War Game.

I. INTRODUCTION

> The enemy of our games was always--Japan--and the courses were so thorough that after the start of WWII--nothing that happened in the Pacific War was strange or unexpected....I credit the Naval War College for such success I achieved in strategy and tactics both in peace and war.
>
> ADM Chester W. Nimitz

Since the early nineteenth century, wargaming has been a part of the military training and planning process. In more recent history, the Naval War College has been the center of U.S. professional wargaming, particularly naval wargaming. The Naval War College War Gaming Department hosts over forty games a year to a wide range of customers. Most of the games are seminar games, although some utilize the aging Enhanced Naval Wargame System (ENWGS).

ENWGS is a computer-based wargame system that models naval warfare to a high degree of fidelity. In addition to somewhat realistic modeling of combat results, a large portion of the "fog of war" is simulated by separating the players from the "game floor," and, to some degree, from each other. In a typical ENWGS game, the players are divided into cells, each tasked with a distinct mission relating to the overall goal of the military forces portrayed. Each cell has a fully functioning remote ENWGS workstation where players and "facilitators" enter the commands that cause the movement and battle of forces in the game. Physically separated from the cells is the "game floor." Here "umpires" have access to the "ground truth" and manage the game situation. They have ultimate control over the unfolding of the story that guides the game results. Thus, a measure of separation of the player from full knowledge of reality is achieved. The result is a wargame that has the feel of uncertainty that closely imitates that of combat command.

Some important elements of the truth are lacking, however. Although ENWGS is very good at modeling combat and command and control in excruciating detail it does not do much to model logistics. The ENWGS model of logistics is very unsophisticated at best. The only logistical consideration in an ENWGS wargame is the monitoring of unit ordnance and fuel levels. There is no modeling of replenishment, force deployment, consumption of other commodities, etc.... If a ship or aircraft (there are no ground units) runs out of fuel or ordnance, the umpire merely replenishes the unit by resetting it to its original levels. If the Naval War College is serious about adding logistics issues to wargaming, then a means to adequately model surge and sustainment must be provided. In a sense what is needed is a "ground truth" generator tasked specifically with modeling logistics.

Unfortunately, the Naval War College does not have access to a sufficiently flexible, user friendly, computer-based logistics model. An informal review of logistics models in NOV 92 revealed that there was not a completely adequate logistics computer model available to the NWC War Game Department (See Appendix A). Although several models provided some of the answers to some of the logistics questions, none of the models covered both surge and sustainment of forces and none of the models were sufficiently robust enough to handle most of the sustainment issues at an appropriate level.

This thesis presents a model, the Surge and Sustainment Simulation, S3, which has the necessary characteristics to allow the Naval War College to add logistical constraints to their wargames, both ENWGS and seminar. First, the relevant characteristics required of such a model will be defined. Then, the Surge and Sustainment Simulation will be described with a view to answering these stated requirements. Finally, an example of a wargame run with the logistics model is provided to demonstrate S3's flexibility and typical operation.

II. LOGISTICS MODEL REQUIREMENTS

The theater logistics problem can be modeled as a network of nodes and arcs through which commodities flow. At a minimum, commodities move from their initial starting points through the network of logistics sites to the units which consume them. Of course, the flow of commodities does not necessarily need to be modeled as discrete shipments, but that would reflect the actual system better than an abstract notion of continuous throughput. Discrete event modeling is especially important if one wishes to determine the state of the system at any given moment. In order tc capture the discrete nature of the movement of commodities from place to place, the arcs of the logistics network can be modeled by individual cargo vehicles which "move" between nodes and "arrive" as discrete events.

A logistics simulation of this type would logically include several different entities. Loosely defined, these are: *commodities*, the items of interest that must be moved from place to place in order to achieve a military objective; *transporters*, the vehicles that are required to move the commodities from place to place; *bases*, the transhipment nodes or logistic support sites through which commodities must pass (such as ports of embarkation or debarkation); and *units*, the end-users of the commodities.

For a logistics model to be effective as a wargaming aid, it should have characteristics which distinguish it from other simulations, just as wargames are distinguished from other simulations. These characteristics can be divided into three categories. They are:

- Wargaming Requirements
- Surge and Sustainment Requirements
- Interface Requirements

Wargaming requirements represent the characteristics of the model demanded by the nature of wargames. Surge and sustainment requirements represent the features of a model whose main function is to describe the state of a theater logistics system. Finally, the interface requirements are the characteristics of a desirable model from a user's standpoint.

A. WARGAMING REQUIREMENTS

One of the main differences between an ordinary simulation and a wargame is the injection of human decision-makers into the feedback loop. As a result, wargames cannot be considered as true analytical tools because repeatability and replication have been sacrificed in order to capture user interaction. Nevertheless, wargaming serves the purposes of training, gives a broad view of the capabilities of forces, and provides clues to aid the decision-maker in the employment of forces. In order to bring logistics into wargaming, one needs a model that describes the logistics situation and reflects the flow of the game.

Obviously, a logistics model designed as a wargaming aid must be driven by game events to some degree. It makes little sense to have a model that is very accurate in predicting the long term, steady state throughput of a logistics network if one is interested in the state of the system at any point in the game. Of course, one will want to know the current state of the system from time to time during the course of a wargame. This is particularly true for umpires who are using a logistics model as substitute for "ground truth." Thus, any logistics model must provide some measure of "real time" logistics information. This implies a model where game events drive inputs and, thereby, affect the output of the model. Two requirements result. First, the model must interact with the user. Second, the model must interact with the game.

1. User Interaction

In this case, user interaction means that the user has the ability to change the characteristics of the logistics network during the execution of the model. Some of those characteristics should include the location and properties of bases, the number and location of transporters, the inventories of units and bases, and the relative priorities of commodities of interest. The user must retain the capability to respond to difficulties arising from a flawed logistics plan of action or unforeseen circumstances such as bottlenecks or changing priorities.

2. Game Interaction

Similar to user interaction, game interaction implies that the model must be responsive to changes in the state of the logistics problem driven by game events. Examples of game events are as obvious as enemy action against logistics sites or transporters, or as subtle as changes in unit consumption rates based on activity level. The complete logistics model must allow these events to affect the state of the system.

B. SURGE AND SUSTAINMENT REQUIREMENTS

At the heart of the model must be the ability to perform calculations and predict the future state of logistics given specific inputs. The surge and sustainment requirements are, in essence, the different logistics functions that the model must perform to meet the needs of the user. These functions are:

- Providing unit closure information
- Providing current supply status
- Replenishing game units
- Tracking unit consumption of commodities
- Tracking throughput of logistics sites
- Handling of the movement of commodities

These functions must be provided for a logistics model to be useful to the wargamer. Preferably, these functions will be performed with minimum intervention by the user.

1. Closure Information

For wargame purposes, Closure Information means the information associated with the arrival of units and equipment in the theater of operations and their movement to final deployed positions. Of interest is the arrival time of units, when some nominal portion of personnel and equipment are in place at desired locations. Additionally, information on the interim supply status of deploying units is desirable.

2. Supply Status

The supply status of a unit or base is essentially its inventory position. Of interest are the amounts of commodities on hand, the amount on order, and the amount on backorder.

3. Replenishment

In order for the model to efficiently depict logistics in the wargame theater of operations, some mechanism must handle the replenishment of commodities for units and bases. For ease of use, replenishment should occur with minimal user intervention.

4. Consumption

Consumption should occur automatically and should be driven by both game events and logistic planning factors. More specifically, logistic planning factors should drive normal daily consumption, while game events, such as raids and attacks, should drive exceptional consumption.

5. Throughput Calculation

As a matter of interest, some measure of the throughput of each base in the logistics pipeline will aid the player and facilitator in fine tuning the logistics system. If bottlenecks or other difficulties arise in the execution of the logistics model, there should be some diagnostic measure to detect these problems.

6. Transportation of Commodities

It is clear that a logistics model must have some means of handling the transportation of commodities. For the purposes of wargaming, the modeling of transportation should be fairly accurate and detailed. The commodities and equipment should arrive at bases and units in discrete packages and at realistic intervals determined by the travel and administrative time required for shipment.

C. INTERFACE REQUIREMENTS

Any computer-based model has certain requirements that are driven by its inherent complexity. Of particular interest is ensuring that model complexity is not a deterrent to effective use. With that in mind, it is important that a wargaming logistics model be menu driven, have an easily used database structure, and have a built-in scenario editor to ease the creation of required data files.

1. Menu Driven

Today, most computer applications are menu driven at the very least. The wargaming logistics model should be no exception. In the best case, the model should provide a windows environment. Of course, issues of compatability with computer operating systems that do not support a windows environment imply that a model prototype may be limited in scope. At a minimum, the model must be menu driven. These menus should be mnemonically-based with commands that have some commonality so that navigation through the various functions of the model is relatively simple and straightforward.

2. Database

In order to relieve the user of the duty of entering specific scenario data every time the model is executed, a built-in database is an important feature. The database must include the following information:

- Logistics planning factors
- Unit characteristics
- Base characteristics
- Transporter characteristics
- Commodity characteristics

Scenario building and database modification should be performed within the program structure without having to resort to the use of an external database program or text editor.

D. SUMMARY

For a logistics model to be effective as a wargame tool, it must meet several minimum requirements. Beyond the obvious requirement that the state of the logistics system must be described in detail, there are other requirements which should be met. These requirements include flexibility in model-user interaction and model-game interaction, a menu driven user interface, a simple but effective database, and a built-in scenario editor. Without fulfilling these requirements, a model will either be unable to provide sufficient information for wargaming or be too difficult to use effectively.

III. MODEL DESCRIPTION

A. OVERVIEW OF THE MODEL

The Surge and Sustainment Simulation is an interactive discrete-event simulation written in MODSIM II and executed on a UNIX workstation. MODSIM II is an object-oriented language especially built to create simulations using the C programming language. S3 is menu-driven and has a built-in *Scenario Editor* to make data entry and scenario generation relatively easy. The model is designed to be executed in coordination with a wargame that does not already possess a satisfactory logistics capability. Execution is interactive as the user follows the events of the wargame, providing the link between game events and the logistics situation.

The principal user of S3 would be a *Logistics Umpire*, a specifically designated member of the game control staff. The Logistics Umpire would act as the communications link between the model and the game, the model and the players, and the model and the umpires. It is the Logistic Umpire's job to see that inputs to S3 follow game events and that the players can incorporate logistics into their decision making process. Additionally, the user will configure the logistics network to reflect player desires as much as possible.

During execution, the model creates a logistics network of transhipment nodes called *Bases* and logistics customers called

Units. Each Base or Unit maintains a unique geographical position (in Latitude and Longitude) that generally corresponds to the actual or proposed location of that Base or Unit. Additionally, each Base or Unit maintains an inventory of commodities, and up to four different types of *Ports* (Airports, Seaports, Rail yards, and Truck stops). A diagram of Base/Unit structure is depicted in Figure 1.

For example, Norfolk, VA could be represented in this model as one Base that maintains a seaport, an airport, a railhead, and truck handling facilities, since all exist within the reasonable confines of Norfolk. Alternatively, one could model Norfolk, VA, as a complex of several bases, each with its own differing types of ports. The decision lies with the user's desire for modeling detail or simplicity.

The Base or Unit inventory contains all of the logistical commodities of interest to the user that are maintained at that Base or Unit. Each *Commodity* in a Base or Unit *Inventory* represents a line item of a typical inventory. Of interest in determining inventory position are the quantity on hand and the quantity on order. Each Commodity also maintains a stocking objective and an order point. These are used to determine the timing of orders or requests for more of a particular Commodity.

Bases and Units are grouped into echelons based on their relative proximity to the theater of operations and their logistical function. These echelons are *Units*, *Theater Bases*,

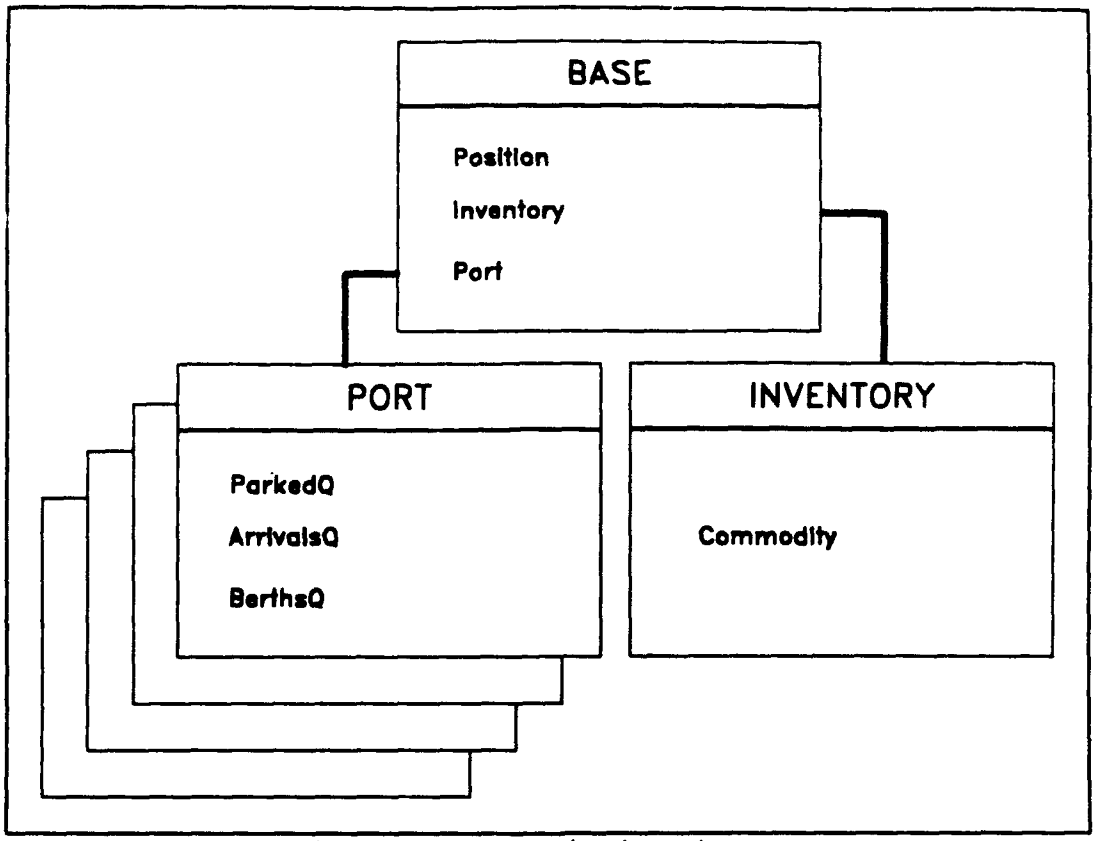


Figure 1 Base/Unit Diagram

Intermediate Locations (ILOCs), CONUS Bases, Ports of Embarkation (POEs), Ports of Debarkation (PODs), and Ports of Supply (POSs). The relationship between Bases and Units is depicted in Figures 2 and 3. All Units are assumed to be in the theater of operations and, therefore, fall in the Units group. Any Base that is "in theater" is normally placed in the Theater Bases group. A Base that is not "in theater," but also not "in CONUS" such as Sigonella or Diego Garcia would be placed in the Intermediate Location group. Of course, any Base in CONUS would be placed in the CONUS group. This should include the origins of all deploying units. For example, if

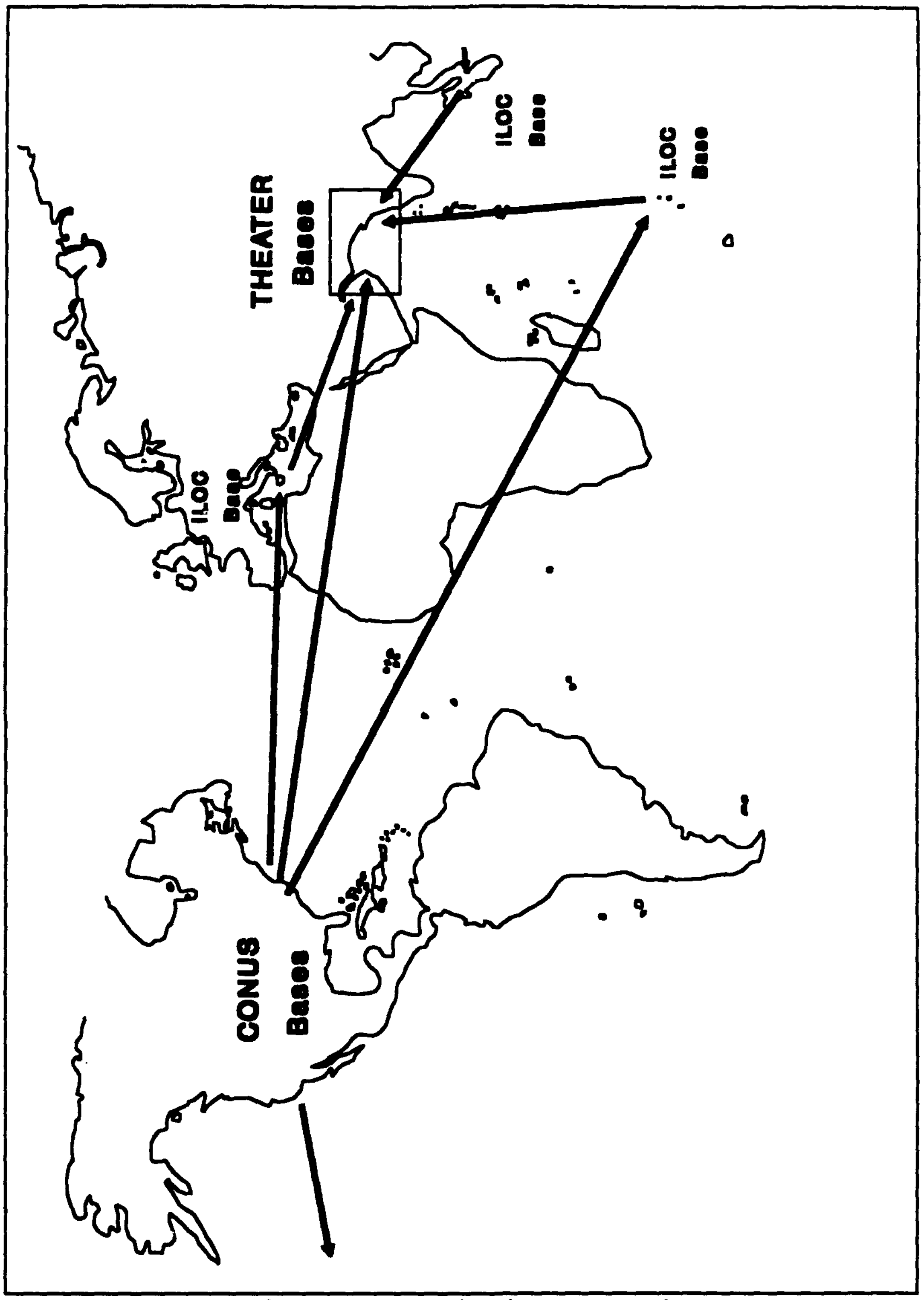


Figure 2 Logistics Network

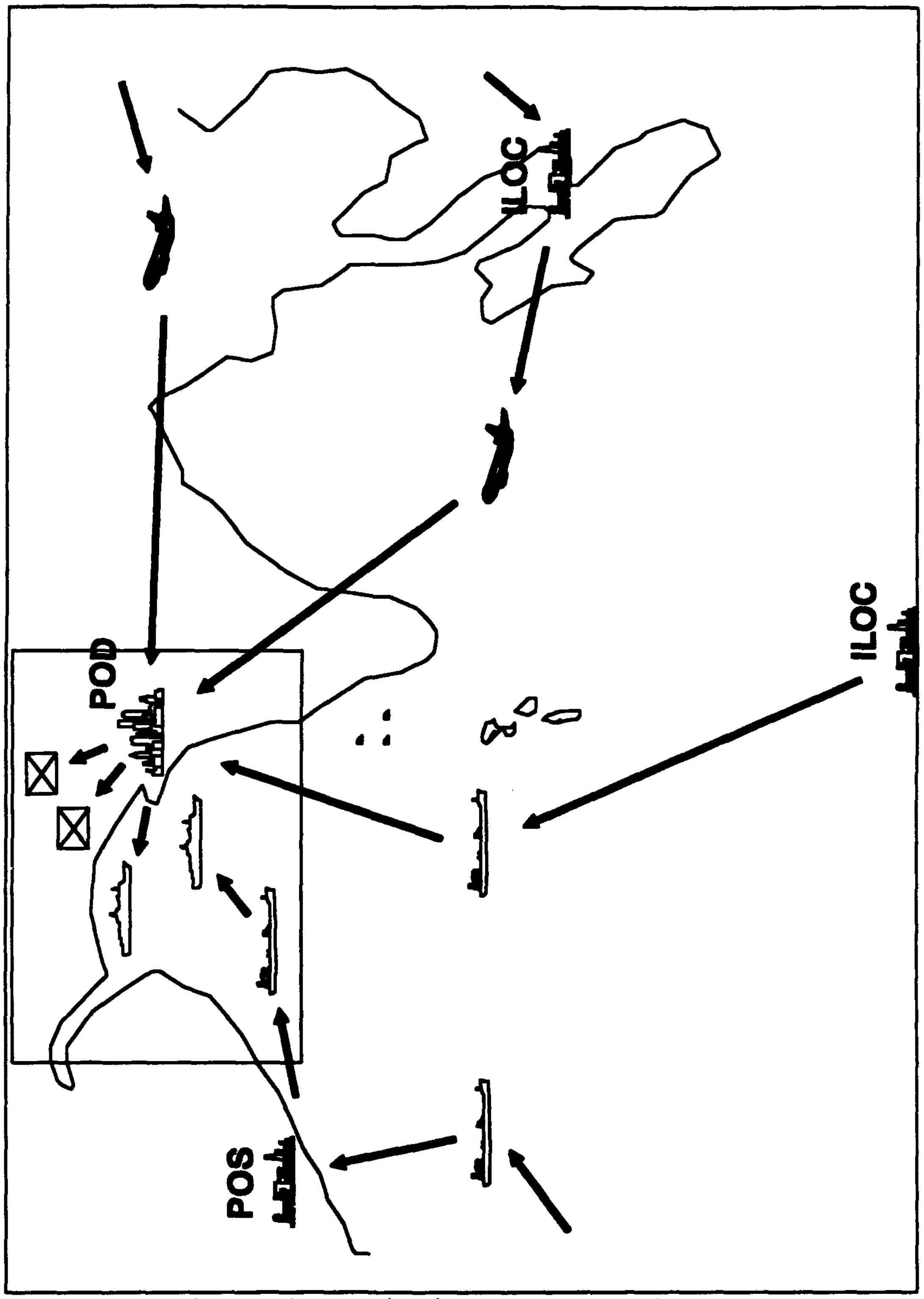


Figure 3 Logistics Network--Theater

one wishes to model the deployment of the 10th Mountain Division, one would place its origin, Ft Drum, NY in the CONUS Base group.

The final three groups are really subsets of the first four. A Port of Embarkation is any Base where Unit equipment or personnel embark on Transporters for movement to the theater during the surge phase of a contingency. In the previous example, the 10th Mountain Division might use MOT Bayonne, NJ as a Port of Embarkation for its equipment while 10th Mountain personnel might use Rome AFB, NY as their Port of Embarkation. Similarly, a Port of Debarkation is any Base that serves as a point of entry to the theater of operations during the surge phase of a contingency. A Port of Supply is a Port where supplies or unit equipment enters the theater of operations during the sustainment phase of the contingency. In the example, the 10th Mountain might fly its personnel to Riyadh in a Persian Gulf scenario. Equipment and supplies might go to Ad Damman for entry into the theater. Note that Bases may be members of one or more of these groupings. For instance, Ad Damman, Saudi Arabia, is a Theater Base, a POD, and a POS. Masirah, Oman would be a Theater Base and a POS for naval units in its vicinity.

Commodities move from Base to Base or Unit on cargo vehicles called *Transporters*. The Transporters also maintain their own inventory of commodities. This inventory is temporary, however, and is known as *Cargo*. Transporters have all the major characteristics of the vehicles they represent including capacity and speed. They can be in one of four states, "Loading," "Enroute," "Unloading," Or "Available."

B. USER INTERFACE

The Surge and Sustainment Simulation is different from other simulations because of the interactive nature of its execution. Since event control is driven by the movement of the simulation clock, user interaction is driven by the clock as well. During execution, the simulation will run for a defined period of simulated time, then query the user for input through the program menu. As a default the program is set to run for 24 time units or one simulated day. Each time unit is defined as the passage of one hour of game time. If the user wishes to change this time step, the user may do so at the beginning of the execution (time 0.0) or at any point of execution at a menu prompt. Thus, if the user does not need to check the state of logistics for an extended period of time, the user can increase the time step to allow for the passage of many simulated days. Or, if the user needs to check the state of the system in six simulated hours from the current simulation time, the user can set the time step to six hours. The program will halt at the end of the six simulated hours and interaction will be possible once more.

When prompted for a command, the user may view the state of the logistics system, view closure information, make modifications, change the time step, continue execution, or quit the scenario. Thus, the requirement that the user may modify the logistics system is satisfied. Between any time step, the user may make certain modifications to the logistics system that will affect its performance.

A user may change the characteristics of some or all of a Base's or Unit's Ports. The user may change the capacity of a Port, add or subtract other Bases from its transportation network, or even turn it on or off to simulate construction or destruction. Moreover, a user may modify the Inventory control of a Base or Unit by directly changing its on-hand amounts, stocking objectives, and order points. At the direction of the user, units may also change position, combat intensity, and closure status the through time step interaction.

To a lesser degree, the user may modify Transporters by adding them to the system at specific Bases or Units to simulate their assignment to the campaign. Additionally, Transporters may be destroyed by the user to simulate their loss through enemy action, mishap or other calamity.

C. WARGAME INTERFACE

Interaction with the wargame is identical to interaction with the user. Since the S3 cannot communicate directly with a wargame system such as ENWGS or even a seminar game, it is up to a user (such as a game umpire) to track critical game events, and apply the results to the simulation. The modification process is the main method for allowing game events to affect model execution.

For example, a battle group must defend itself from large scale air raids during a particular wargame. Now, the default consumption rates for the battle group, as defined by the user in the Scenario Editor, allows for normal daily consumption of specific commodities. The use of a daily rate is highly inappropriate for consumption of threat ordnance such as surface-to-air missiles (SAMs). Thus, the user should remove a specific number of SAMs from the battle group inventory on a one-time basis to represent the shock to the logistics system of the expenditure of a large number of SAMs. Once the on-hand quantity of SAMs has been reduced, the battle group will order the commodity during its next inventory check. Resupply timing will be determined by the normal execution of the program. By closely observing the state of the battle group at various points in time following the raid, the user can ascertain when the battle group has received sufficient SAMs to permit further action.

Of course there are other ways to simulate important game events. Unit engagements can be simulated by raising their combat intensity level. Currently, the S3 models daily consumption as a function of combat intensity. There are four specific combat intensity states (High, Med, Low, and None) for a Unit. This is a compromise between the highly detailed consumption model, such as found in U.S. Army publications, and a simpler system such as planning factors based on average consumption. The highly detailed alternative implies long hours of data input, while the other alternative implies unrealistic consumption rates that are quite unrelated to combat. Nonetheless, the user retains some degree of control over the consumption rates of Units without resorting to directly changing rates by hand. With this mechanism, the user can represent the beginning of a new phase of a campaign by increasing the combat intensity of engaged units.

Enemy action against the logistics pipeline itself may be simulated by selective destruction of Bases, Ports, Commodities, and Transporters in the modification process.

D. OPERATION OF THE MODEL - SUSTAINMENT

Although logically surge comes before sustainment, the operation of the model in sustainment will be discussed first. This is done because the concepts involved in sustainment also apply to surge and will be easier to present initially, saving the exceptions of the surge case for later.

The model depicts the day to day supply state of Units and Bases in a wargame by simulating the consumption, requisition, movement and production of Commodities. For the purposes of the simulation, each unit of simulation time is equivalent to one hour of "real" time. Every 24 hours, each Base or Unit iterates through its inventory and performs operations on the Commodities it finds (Figure 4). Each Unit consumes its

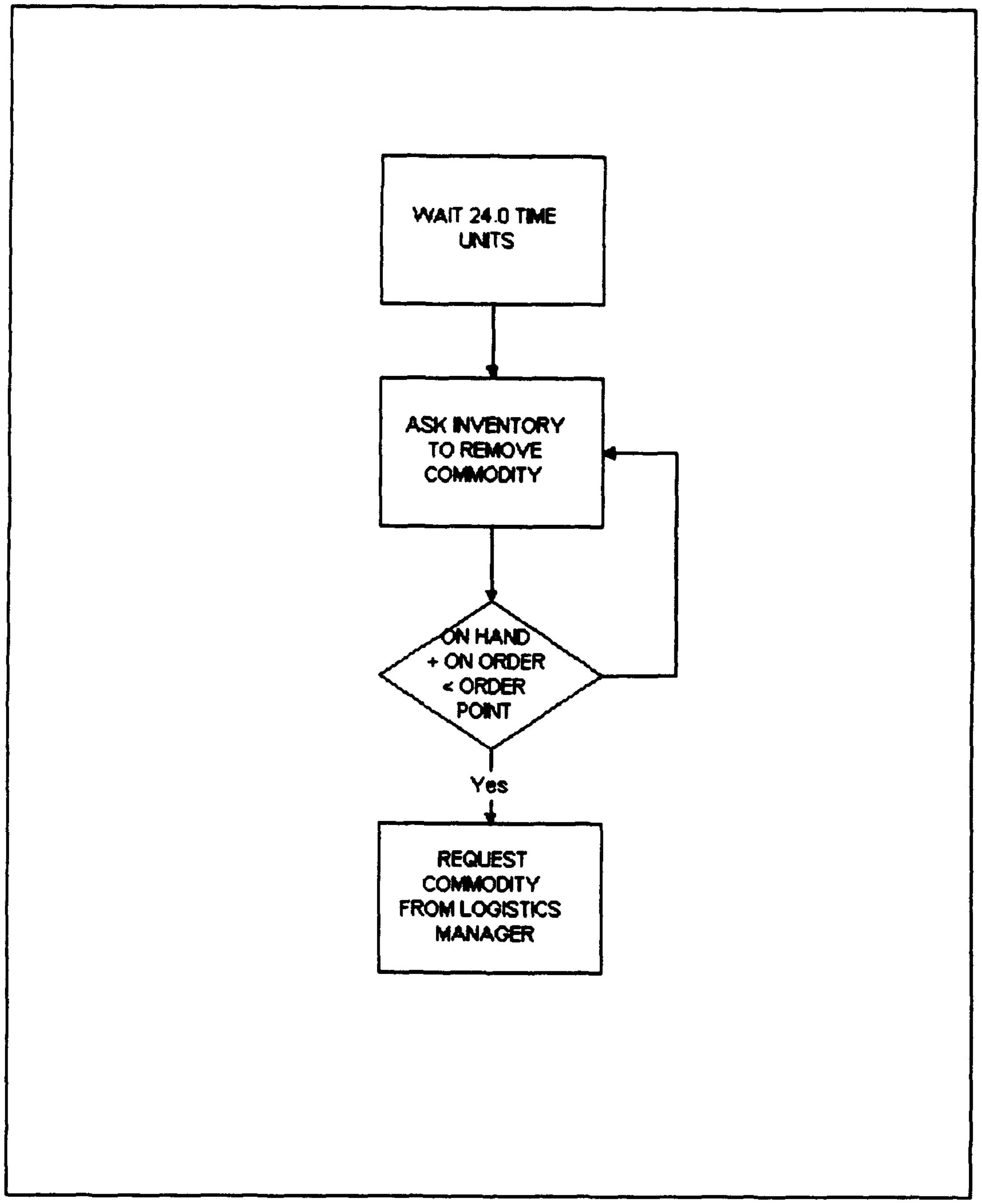

WAIT 24.0 TIME UNITS
ASK INVENTORY TO REMOVE COMMODITY
ON HAND + ON ORDER < ORDER POINT
Yes
REQUEST COMMODITY FROM LOGISTICS MANAGER

Figure 4 Base/Unit Check Inventory Process

Commodities by subtracting from its on-hand quantity an amount equal to its specified daily consumption rate. This consumption rate is based on combat intensity and is unique for each Unit and Commodity. The consumption rates are determined by the user when the Unit is created in the Scenario Builder. Then, if the on-hand quantity for that Commodity is less than or equal to its order point, the Unit places an order with the Logistics Manager. The same process is performed by Bases, except that Bases do not consume Commodities.

1. The Logistics Manager

When a Commodity shortfall is identified, the Base or Unit passes this information to the Logistics Manager (Figure 5). The Logistics Manager determines the best supplier of the Commodity. This is usually defined as the closest Base in the next highest echelon that has the desired Commodity in its inventory. For example, if a Unit places an order, the Logistics Manager searches for the Commodity among the Theater bases first, then, if the order is not satisfied, among the Intermediate Locations. If the order is still unsatisfied, the Logistics Manager searches through the CONUS bases. Finally, The Logistics Manager will place an order with the closest CONUS base, which will place the request on back order. There is one major exception to this search routine. If a Unit is requesting a Commodity that it has flagged as a

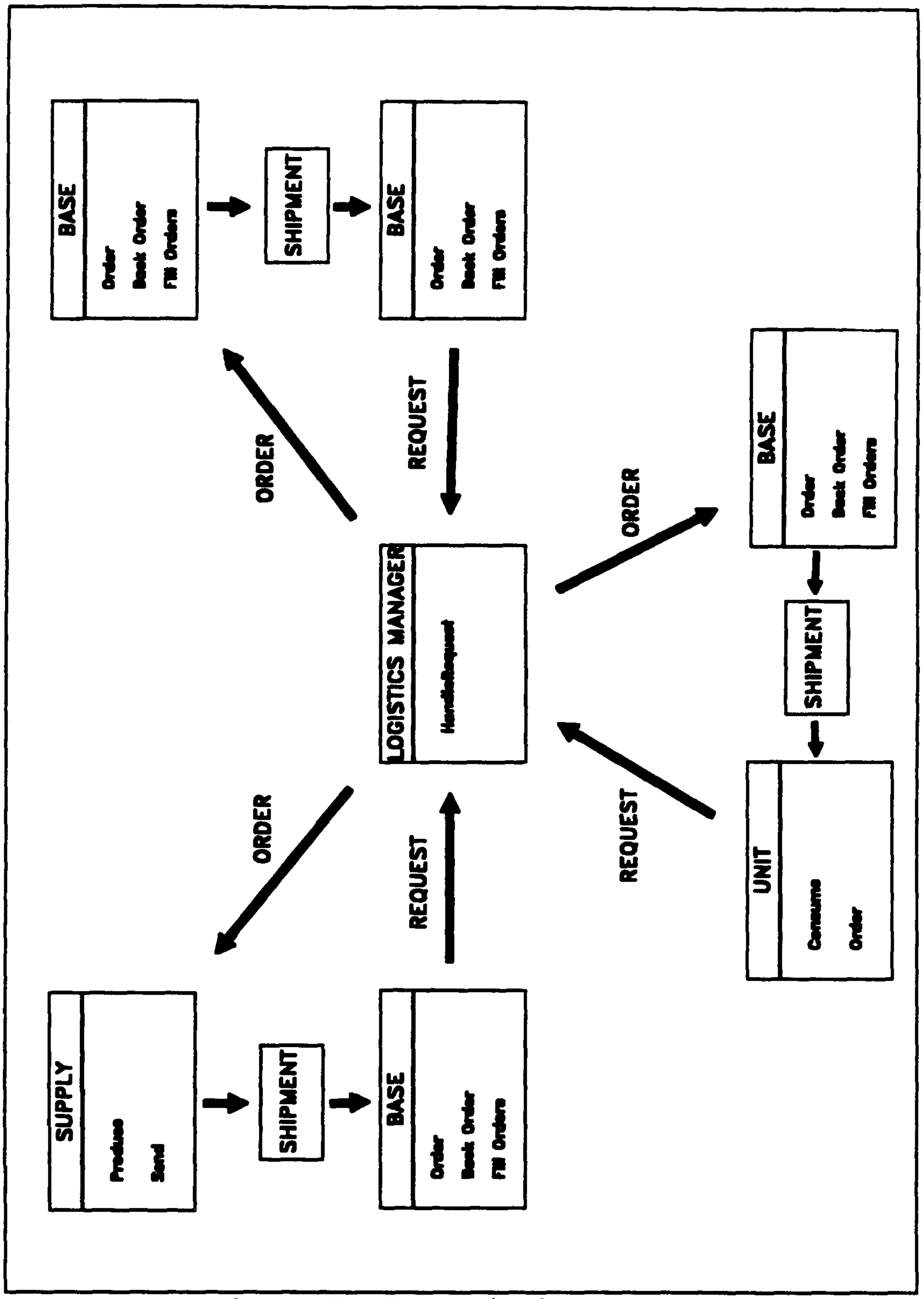


Figure 5 The Logistics Manager

Deployment Commodity, and the Unit is not yet considered *deployed*, the Logistics Manager will go directly to the Unit origin as defined by the user in the Unit creation process.

Next, the Logistics Manager chooses the routing that the Commodity will follow on its way to the customer. For Commodities ordered from a Unit origin during its surge phase, the route is from the Origin to the closest appropriate POE, to an appropriate POD, then to the Unit. The closest appropriate POE or POD is determined by distance and Commodity Priority.

Each Commodity has a unique priority assigned by the user. Items with priority one through three will be sent by air whenever possible. If air transport is unavailable, then the commodity will be sent by rail, then truck, and finally ship. Items with priority four through six first go by rail, if feasible, then truck, ship, or air. This logic continues for priorities seven to twelve. In the case of POE to POD or POS travel, the Logistics Manager will restrict its search to air and sea travel. In the surge case, the Logistics Manager will check the POD list first, looking for the closest POD to the Unit that has the appropriate Port (Air or Sea). Given that the POD has an airport or seaport, the Logistics Manager will then choose the appropriate POE that is closest to the Unit Origin with similar port facilities.

As an example, assume a theater that has no serviceable airfield to act as a POD for priority one shipments. The Logistics Manager would be forced to send surge shipments to the nearest seaport to the requesting Unit. This will force the Logistics Manager to choose the closest seaport to the supplying Origin even if the closest POE is an airfield. In this way, Commodities will not be sent to CONUS Bases that cannot communicate with Theater Bases.

If a Base or a Unit in its sustainment phase places an order, the Logistics Manager creates an alternative routing. For sustainment routing, the Commodity will travel from the best CONUS supplier to the nearest appropriate POE to the nearest appropriate POS and then the Unit or Base. Of course, if the customer is a Base at an Intermediate Location, the Commodity will travel directly to the Base rather than a POS. By the same token, if the best supplier is a Theater Base, the Commodity will be moved directly to the Unit.

After the route is built, the Logistics Manager will create an Order into which the Commodity information, the routing, and a reference to the Destination are placed. This Order is passed to the selected supplying Base, which processes the request. The process is depicted in Figure 6.

2. The Supplying Bases

The supplying Base receives the Order and checks its current inventory. If there are sufficient quantities of the Commodity on-hand, the Base will decrement the requested quantity from its inventory and send the filled Order, or *Shipment*, to the appropriate Port for transportation. If

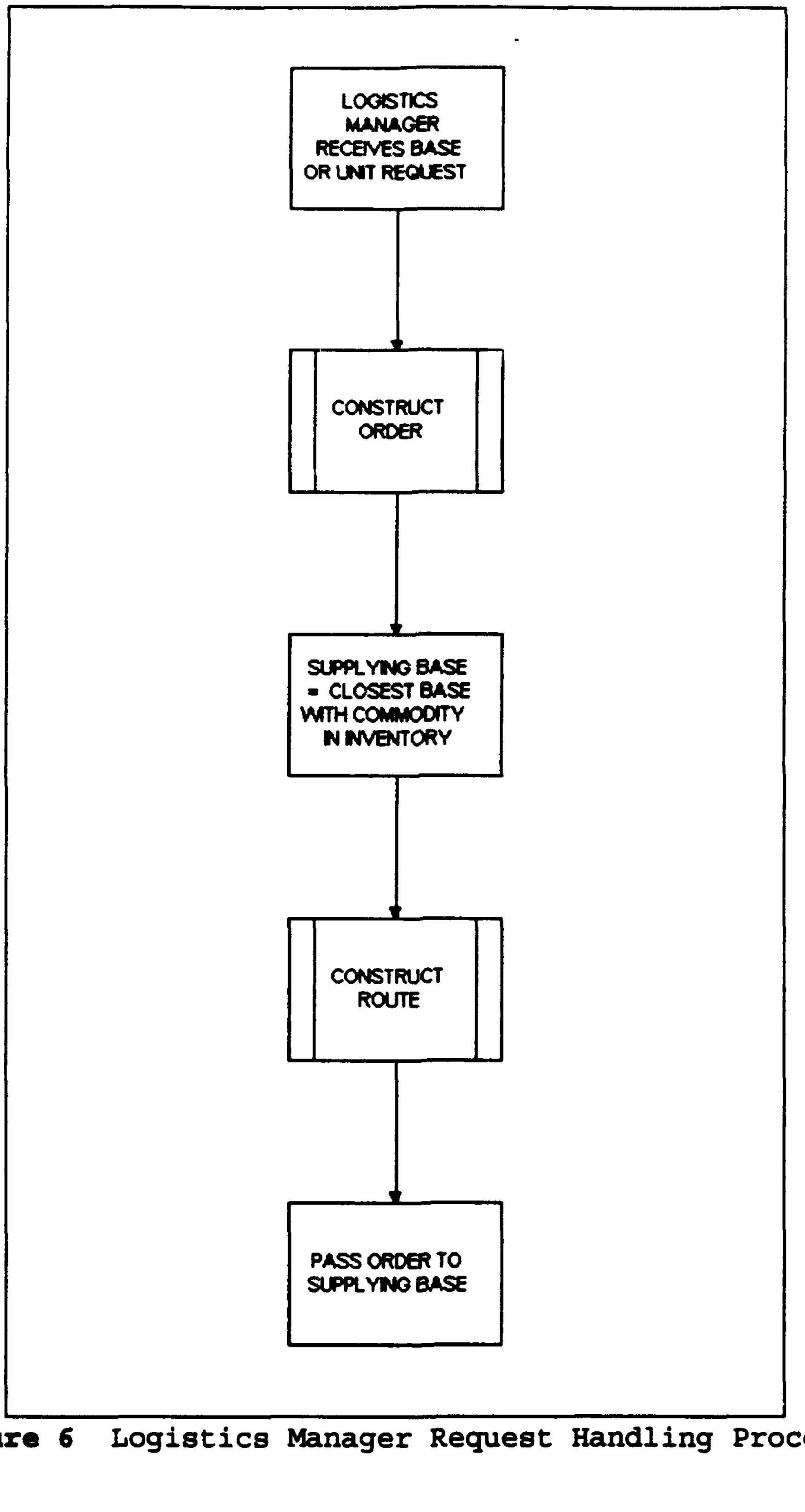

LOGISTICS MANAGER RECEIVES BASE OR UNIT REQUEST
CONSTRUCT ORDER
SUPPLYING BASE = CLOSEST BASE WITH COMMODITY IN INVENTORY
CONSTRUCT ROUTE
PASS ORDER TO SUPPLYING BASE

Figure 6 Logistics Manager Request Handling Process

there are insufficient quantities, the supplying Base will send the quantity of the Commodity that it has, then place the remainder on back order. Whenever that Base receives a Shipment from a supplier, it will check its back orders to decide if it may fill an Order.

As described previously, routing is based on Commodity Priority. This is true of Port selection, as well. A Base will select the fastest possible means of transportation based on the priority of the Commodity, the types of Ports serviceable at the immediate destination, and the types of its own serviceable Ports.

Once the Shipment is sent to the appropriate Port, the Port will sort it into a *Cargo Group* based on destination. Each Port has two *Loading Docks*, an *Outsized Loading Dock*, and a *Normal Loading Dock*. As the name implies, the Outsized Loading Dock holds only Cargo Groups that contain Shipments of outsized Commodities. Here, an outsized Commodity is defined as a Commodity whose dimensions exceed the dimensions of the C-141B cabin area (the standard Air Force definition).

Although outsized items are normally defined for the purposes of air transportation, the concept is used throughout the program to preserve the generality of program constructs, and, to enable the user to model special Commodities and special Transporters. An example might be the movement of Armored Fighting Vehicle (AFVs) by truck. Clearly, these Commodities would be outsized for air transport, but, they would also require Heavy Equipment Transporters (HETs) for long-distance over land movement. The user would need to create Transporters that could move outsized Cargo over land as well.

3. Transporters and Cargo Movement

When an outsized transporter arrives for loading, such as a C-5A Galaxy aircraft, it will begin loading the highest priority Shipment on either Loading Dock. After the first Shipment is loaded, the transporter will load any Shipments bound for the same destination by priority, alternating between the Outsized and Normal Loading Dock. This process stops when the transporter has reached maximum capacity (by cube, area, weight etc...) or the Cargo Groups for that destination are empty. Once the transporter is fully loaded, it departs for the next Base or Unit as defined by the destination of the highest priority Shipment.

Movement of transporters from place to place is handled in the most straight- forward and simplified manner. The Great Circle distance is calculated between the departure point and the destination utilizing their defined positions given in latitude and longitude. The distance is simply divided by the user defined cruise speed of the transporter to arrive at the travel time. After "waiting" for this travel time, the transporter "arrives" at its destination and unloads its cargo.

When the Transporter arrives at a Base or Unit, the Base or Unit sends the Transporter to the appropriate Port. All Ports have three queues; one for arriving Transporters, called the Arrival Queue, one for loading or unloading Transporters, called the Berth Queue, and one for Transporters that are finished unloading, called the Parked Queue. An arriving Transporter is sent to the Arrival Queue until space is made available at the Berth Queue by a departing Transporter. Once it finishes unloading, the newly arrived Transporter loads any Shipments waiting for transportation to other destinations. If there are none, the Transporter moves to the Parked Queue and makes itself available to the *Transporter Manager* for assignment.

Shipments that are unloaded at a Base are sent to other destinations based on their routes and priorities. This process continues until the Shipment reaches its final destination where it is added to the customer Inventory. This process of ordering, supplying, transporting and consuming continues, driven only by the passage of simulated time.

4. The Transporter Manager

The Transporter Manager (Figure 7) is a program construct that is very similar in nature to the Logistics Manager. The key functions of the Transporter Manager are to track available Transporters, and, to send them to requesting Ports. As a result, the Transporter Manager maintains a queue of each Transporter and each request by Class.

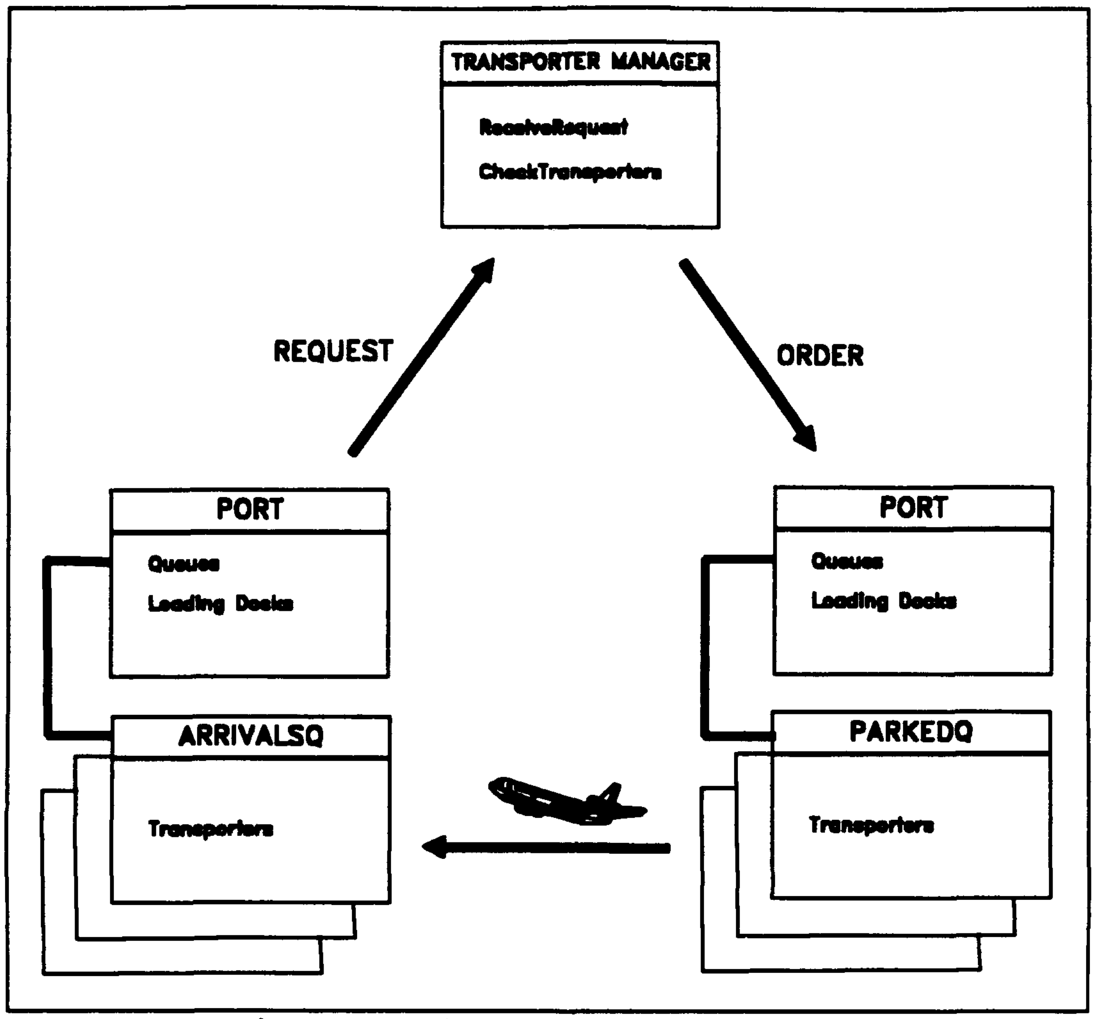


Figure 7 The Transporter Manager

When a Port creates a new Cargo Group (i.e., it sends a Shipment to a new destination), it sends a request to the Transporter Manager. This request includes a reference to the requester, the Class of Transporter requested, the SubClass of the Transporter requested, and a specification that the Transporter must handle outsized cargo, if necessary. The Cargo Group keeps track of the number of Transporters on request by tracking an estimate of the nominal cargo capacity that has been requested. When the load size of a Cargo Group exceeds the nominal capacity already on request, the Cargo Group requests additional Transporters through its Port. When a Transporter removes Cargo from the Cargo Group, the Cargo Group subtracts the size of the load transported from the estimate of Transporter capacity on request. Thus, requests for Transporters are directly and automatically driven by the load size and the number of Transporters already on request.

If the Transporter Manager receives a request, it immediately checks the Transporter queue corresponding to the requested Class. The Transporter Manager will send the closest available Transporter that meets the requested Class, SubClass and outsize specifications. If no eligible Transporter is found, the Transporter Manager will add the new request to a list of old requests to be filled when the proper Transporter becomes available. When the Transporter Manager receives an available Transporter, it will search through its request list to determine if the new transporter meets the needs of any old request. If it does, the Transporter will be sent to the requesting Base to pick up Cargo. If the Transporter does not fit a requested need, it is placed in the queue of available Transporter on call for a new request. The processing of requests is accomplished on a "first-in, first-out" basis. This process is summarized in Figure 8.

E. OPERATION OF THE MODEL - SURGE

Unit deployment is modeled in much the same way that sustainment issues are handled. The same order/routing

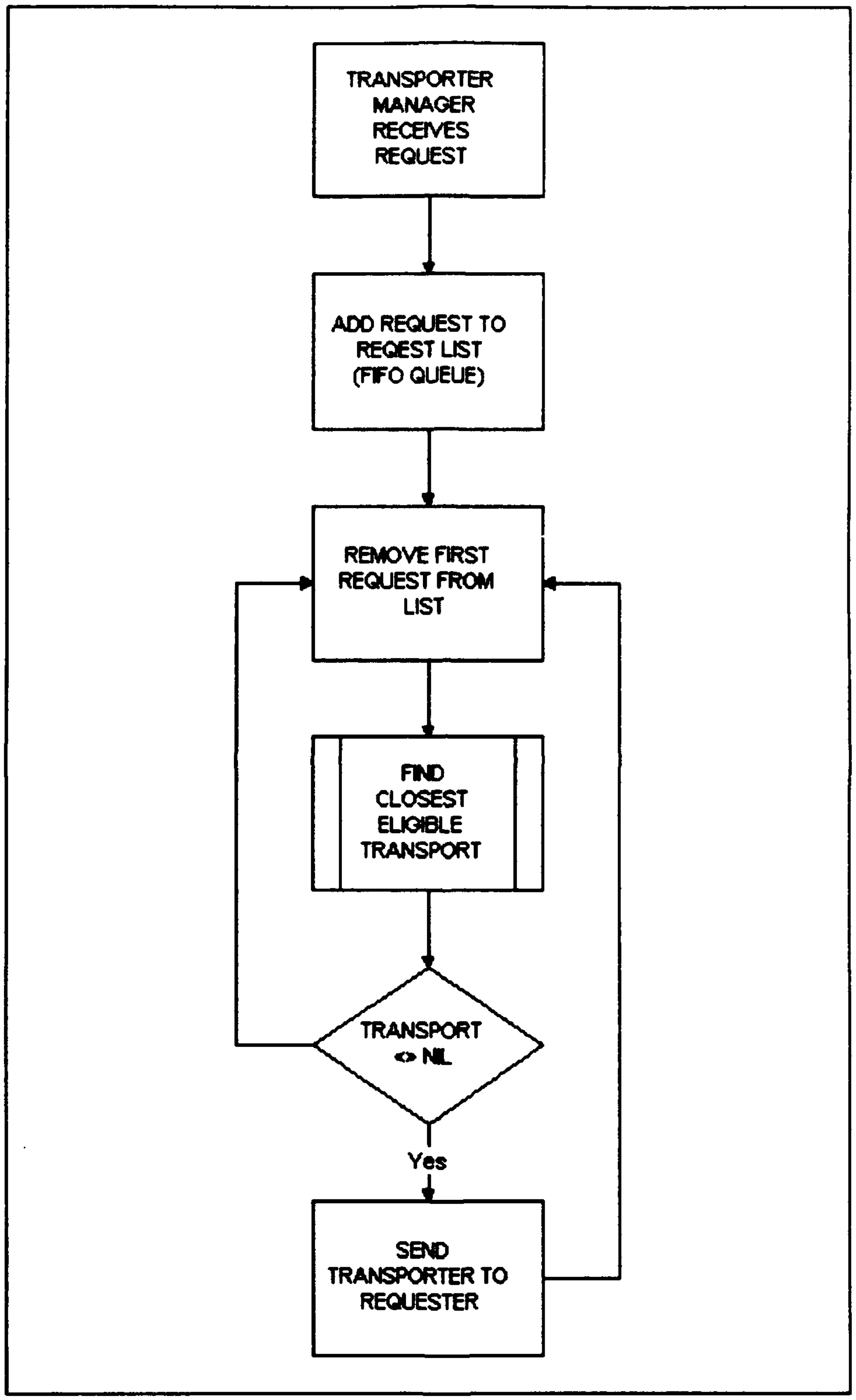

TRANSPORTER MANAGER RECEIVES REQUEST
ADD REQUEST TO REQEST LIST (FIFO QUEUE)
REMOVE FIRST REQUEST FROM LIST
FIND CLOSEST ELIGIBLE TRANSPORT
TRANSPORT <> NIL
Yes
SEND TRANSPORTER TO REQUESTER

Figure 8 Transporter Manager Request Handling Process

mechanism is utilized for deploying Units as those already in the field. It takes little imagination to realize that this is a reasonable means of dealing with what seems to be two entirely different problems. Since the actual physicalactions taken in deploying a combat unit to theater are very similar to sending a shipment of some commodity, this makes intuitive sense. Essentially, when a Unit deploys, it requests that its equipment, personnel, and supplies be moved to its deployed position. So, the Unit represented by the model is merely the location to which the equipment, supplies and personnel will be sent. In effect, a Unit is an empty box that is waiting to be filled. Only when a specified amount of certain commodities arrives at the box will the Unit be considered deployed.

For example, a Unit, such as a heavy Tank Battalion, is attempting to deploy to the game theater. When the user created the Unit with the Scenario Builder, the user specified the Commodities that the Tank Battalion would maintain in its Inventory, based on the actual Unit TO&E. At this time the user also specified which Commodities count toward the deployment of the Unit. For instance, the user may specify that the M-1A1 Main Battle Tanks, Personnel, 120mm APDS and 120mm HEAT ammunition are all Commodities that must be received by the Unit before it is considered deployed. Moreover, the user specified that 85% of these Commodities must arrive before the unit is "deployed." When the simulation begins execution, the Tank Battalion sends a request for the tanks, personnel, and ammunition in order to start the deployment process. Deployment timing will be determined by the phasing of these requests with requests from other Units attempting to deploy at the same time. To space these Unit requests, the user may specify a time delay for the Unit before it mobilizes. This time delay is a means of ordering the deployment of Units in the game without resort to building Time-Phased Force Deployment Data (TPFDD).

What this method of modeling deployment avoids is the requirement to build a Transportation Feasibility Assessor that is based on a TPFDD. Strategic deployments really are handled in this way in reality, but this represents an increase in programming complexity in orders of magnitude. Clearly, a TPFDD builder is not appropriate to the simple demands of wargaming.

In any event, the model does handle deployment, albeit in a more primitive manner than the real world. The deployment status of the operation can be displayed in one of two ways. First, the user may display the aggregate on-hand level of *deployment* Commodities for all game Units from the Time Step Menu. The proportion of all deployment commodities to their stocking objective is displayed by category for all Units in the scenario. Second, the Time Step Menu displays similar supply information for <u>all</u> commodities in Unit inventories. This second display is the default selection and is automatically displayed at every time step. Thus, an easily understood display of aggregate supply status is available whenever the user is prompted for a command.

F. THE SCENARIO EDITOR

The Scenario Editor was created to handle the tremendous amount of data required to simulate the logistics of a typical wargame scenario. With it the user can build the necessary Units, Bases, Transporters and Commodities. Essentially, the Scenario Editor creates an instance of the object being built and fills its fields with data obtained interactively from the user. Later, the object is saved to a formatted data file, which is read during execution.

To take full advantage of the object orientation of MODSIM II when the program executes, it uses the data files to create a *Master* object for each different type of object. During execution, object instances are mere clones of the original Master object. For example, rather that repetitively creating 20 separate and identical C-5A Transporter objects, the Scenario Editor builds one Master C-5A data file. At execution, this file is read and the Master C-5A object is placed in a Master Transporter Queue for storage. The 20 other C-5As are mere copies of the original. They are distributed among the Bases and Units according to user instructions contained in Base or Unit data files. The same process is used for Commodities and to a lesser extent, Units and Bases.

Bases are created in a similar manner as Transporters and Commodities. Their names, locations, and port data are recorded through interaction with the user. The type and number of transporters are also determined from the list of existing Master transporters. Thus, one could define Norfolk, VA to begin a scenario with four C-5A aircraft, three breakbulk cargo ships, and two 40 car freight trains. The same type of process is used to pick the Commodities for a particular base.

Creating Units, on the other hand, is very similar to creating Bases. The exception lies in the creation of Inventory. To provide a reasonable estimate of Unit consumption, Unit Inventory is determined by the composition of the Unit. During the creation process, the user builds a Unit by picking from a number of previously built SubUnits, each possessing inventories and consumption rates. These SubUnits are also built with the Scenario Editor. This is done to ease the research effort by the user. For example, a CVBG consists of a CVN, an AOE, and several escorts. Rather than compelling the user to model each individual ship or aggregating the collective inventory of the battle group, the Scenario Editor performs this automatically. The user merely picks the major elements of the Battle Group from the list of previously built SubUnits. Thus, the user could select a Roosevelt CVN, a Supply class AOE, three DD-963s, two FFG-7s, 24 F-14s, 10 A-6Es, etc.... The Scenario Editor will aggregate these SubUnits and calculate an Inventory of the appropriate depth and range with consumption rates in order to create one Battle Group object. The result is flexible creation of Units such as battle groups, which must be tailored to the mission rather than formally defined by TO&E.

After all the necessary Commodities, Transporters, Bases and Units are built, the user must create a scenario. A scenario is defined by the Bases and Units in a given operation and their relationship to each other. Since the number and position of Transporters and the Inventories of Bases and Units are enumerated in the Base and Unit data files, this is all that is required to define a scenario.

G. SUMMARY

As discussed, the S3 model meets all of the requirements of theater logistics simulation designed as a wargame aid. Provisions have been made to allow for flexibility in user-model interface and game-model interface. S3 is menu-driven and handles requirements of simulating a theater logistic problem. It provides an easily understood display of the theater supply status and detailed displays Unit, Base, Commodity, or Transporter status for any scenario. Finally, it features a built-in Scenario Editor to allow the creation of a logistics database and relieves the user of the complexity of building a successful scenario. Demonstrating the capabilities of the S3 through an example of its execution using an actual wargame scenario will now be addressed.

IV. BUILDING SCENARIOS

To simulate theater logistics using S3, the following information must be obtained before building a scenario:

- Commodities: Determined by characteristics of Blue Forces and by player interest
- Transporters: As dictated by scenario, or game administrators
- SubUnits: Individual ships, aircraft, ground units
- Units: Determined by the scenario, aggregates of the SubUnits
- Bases: Determined by the scenario geography and player decision

Once this information is obtained, the scenario may be constructed utilizing the Scenario Editor.

The construction of a scenario must proceed in a specific order since the construction of many of the entities depends, to some degree, on the previous construction of others. It is best to first construct the Transporters and the Commodities of interest. The Subunits which consume the Commodities are then constructed. Units can then be built from the Subunits, Commodities and Transporters. Bases, which require Commodities and Transporters, are built. Any *Prepositioned Transporters (Prepo)* may be assigned locations and Cargo. Finally, the Scenario, which consists of Bases and Units and the various transportation nets, is created.

A. COMMODITIES

1. Construction

Commodities have several qualities that must be defined during their construction. They are:

- Name
- Class
- Production Rate
- Dimensions
- Priority.

Name is the nomenclature of the Commodity. There are few restrictions on a Commodity Name save that it is unique. For practicality, it is best to restrict Commodity Names to less that 15 characters for the purposes of screen display, but there is no real restriction in the model.

Class deserves some attention. Class is the means by which S3 differentiates the loading characteristics of Commodities. Additionally, other functions, such as displaying supply status, are accomplished on a Class by Class basis. The defined classes for Commodities were picked to mirror the DOD "Classes of Supply" system with some exceptions. The classes are:

- Fuel: POL or any liquid transported in bulk
- FFV: Fresh Fruits and Vegetables or any other rations
- Ammo: Munitions of all kinds
- Spares: Parts, maintenance items and other consumables
- Personnel
- Medical: Medical supplies
- Major: Major end items, such as vehicles or large equipment
- Other.

Production Rate deserves little discussion except to say that it represents national production of a particular commodity or the diversion of that commodity from other areas not explicitly modeled. In the S3 model, production rate is the amount that the Supply object produces on a daily basis (24.0 simulation hours).

Dimensions and weight are also obvious characteristics. Commodity dimensions are given in inches and Commodity weights are measured in pounds.

Priority, as discussed in Section III, determines the transportation mode of a particular commodity. Valid inputs are integers from one to twelve. As an example, items with priorities from one through three are sent by air if available at both origin and destination. Alternate means of transportation are selected based on speed. Thus, if shipment by air is not possible, priority one through three items are sent by rail, then truck, and finally, ship. See Table I for a summary of priorities and their respective shipping methods.

Table I - PRIORITY AND METHOD OF SHIPMENT

Priority	Methods
1-3	Air, Rail, Truck, Ship
4-6	Rail, Truck, Ship, Air
6-9	Truck, Ship, Rail, Air
10-12	Ship, Truck, Rail, Air

2. Modeling

From the discussion of Sections II and III, it is clear that flexibility with regard to scenario building is an important facet of the S3 model. Commodities lie at the heart of the model. Their movement and inventory determine the success of the model at accurately portraying theater logistics. Thus, a large degree of flexibility is built-in to the construction and modeling of commodities.

First, Commodities may represent any item the user wishes. The user may be interested in the movement of only the most general items. In that case, the user may wish to aggregate all items of a particular supply class into one commodity. In other words, one could create a Commodity called "Munitions" that would represent a pallet-sized shipment of any form of ammunition. If this level of detail is insufficient, the user could go as far as modeling individual items and rounds. Here, it would be wise to aggregate the other classes and items that are not of interest into one or several other Commodities. By causing the Commodities of interest to compete with these other items, one can offset the unrealistic tendency for the model to deliver Commodities of interest faster than would normally be expected.

Second, some items may be modeled as both Commodities and SubUnits. For example, a HS detachment may be comprised of six SH-60F helicopters. The detachment would be represented as a Unit that was constructed from six SH-60F SubUnits. To simulate the movement of replacement helicopters through the logistics system, it would be necessary to construct an SH-60F Commodity, as well. The HS Unit would maintain six SH-60F Commodities in its Inventory, beyond having been constructed from SH-60F SubUnits. Although the distinction is less than clear, it is important for one to keep in mind that SubUnits are created to ease in Unit construction. SubUnits are not Commodities, and cannot be moved through the logistics network.

B. TRANSPORTERS

Like constructing Commodities, constructing Transporters with the Scenario Editor involves entering Transporter characteristics interactively. The characteristics for each Transporter will be common to all Transporters of the same type that are created at run time. The characteristics that define Transporters are:

- Name
- Class
- SubClass
- Speed
- Range
- Size
- Cargo Area
- Cargo Cube
- Cargo Weight
- Largest Single Item Dimensions
- Passenger Capacity
- Liquid Capacity.

Like the Commodity Name, a Transporter Name can be of any length but is preferably less than 15 characters. It is important that each different type of Transporter has a unique name. When Transporter objects are created by the program, they are assigned individual identification numbers to tell them apart.

Like Commodities, Transporters have Classes, which have been previously described. These Classes are Aircraft, Rail, Truck, and Ship.

Transporters have a SubClass, also previously described. The SubClass is used to characterize the Transporter loading method and primary cargo. The Transporter SubClasses are:

- Liquid: Liquid cargo Transporters (eg. Tankers)
- RoRo: Roll-on, roll-off cargo and major end items
- BreakBulk: All cargo types
- General: All cargo types except liquids
- Pax: Passenger cargo and limited breakbulk cargo.

Transporter Speed represents the cruising speed of the Transporter. In general, the user will enter whatever planning factor is available for the specific vehicle. For example, although most trucks can move in excess of 45 mph, the planning factor defined by the Army FM-101-10 manual is 15 mph. The appropriate action would be to utilize the 15 mph planning factor to account for delays and characterize the typical performance of ground transportation.

Although the same logic would be applied to Transporter Range, the program does not take Range into account. At present, Range information has been provided but is unused by the program. Transporter Range limitations are planned for refinements to the model at a later date.

Transporter Size represents the overall dimensions of the Transporter in whatever units are appropriate to the

Transporter Class. In the case of aircraft, the Transporter Size represents the area footprint or ramp space requirement in square feet. For rail, it represents the number of cars in a train. Transporter Size, like Range, is currently not in use by the model. It is envisioned that in the future the Ports will only accept vehicles below their maximum size, and the Transporter Manager will only choose appropriately sized vehicles in filling requests.

Cargo Area, Cube, and Weight are all limiting factors in the performance of Transporters. Unlike Size and Range, these values are in use by S3 and form the basis for Cargo loading. For the most part, these values are available for all cargo vehicles. The Area and Cube can normally be calculated from the dimensions of the vehicle's usable cargo space. Area and Cube are to be given in square and cubic feet, respectively. Weight is to be entered in pounds. Any stowage factors should be applied before values are input to the model.

Largest Single Item Dimensions normally apply to aircraft, but their use has been extended for the sake of generality among Transporters. Of course, these dimensions represent the outside dimensions of the largest single item a vehicle may carry. This is usually restricted by door and hatchway size, but sometimes there are other factors. Typically, this restriction applies to aircraft. For all other vehicles it may be useful to set these numbers equal or greater than the largest item in the scenario.

Since liquid cargo and passengers are not generally loaded in the same manner as breakbulk or roll-on, roll-off cargo, Transporters have a separate Liquid and Passenger Capacity. For liquids, this represents the maximum usable liquid storage capacity for a vehicle. This should be expressed in terms of 42 gallon barrels (bbl's). Transporters load Fuel Class commodities based on this barrel capacity and ignore all other restrictions. Passengers are handled similarly with the units of measure being individual Personnel Class Commodities. The user should consider cargo aircraft such as the C-5A Galaxy when designing Transporters. Since this aircraft has room for over 70 passengers in the upper cabin area, these passengers normally do not count against cargo restrictions. On the other hand, there are aircraft, like the C-141B, that can carry passengers by sacrificing cargo space. The user is cautioned that it is probably inappropriate to allow a C-141B to haul its maximum cargo and passenger capability simultaneously. A compromise might be to design a C-141B in the C-2 configuration for floor loaded and palletized cargo with minimal passenger capability and a C-141B in a passenger-only configuration. Both these Transporters could be available in a scenario. This technique is fairly realistic since C-141 aircraft often take a long time to reconfigure.

C. SUBUNITS

SubUnits were created to aid in the construction of Units. SubUnits embody all of the consumption planning factors required to simulate theater logistics. In essence, the SubUnit is a list of Commodities and their consumption rates. The SubUnits themselves represent the building blocks of a Unit just as companies are the building blocks of battalions and divisions are the building blocks of corps. If one can visualize a typical unit, such as a carrier battle group, then a logical SubUnit would be an individual ship or aircraft. It is the consumption rates of these individual assets that are depicted by SubUnits. From the individual SubUnit consumption rates, the aggregate Unit consumption rates are derived.

1. Construction

When constructing a SubUnit, the user interactively enters its characteristics in a manner similar to building Commodities or Transporters. Unlike the Commodities and Transporters, the SubUnit is constructed primarily by choosing various items from the list of previously built commodities. Consequently, it is important to build as many Commodities as possible before constructing SubUnits.

Of course, the SubUnit has a Name and a Class. Like all the other constructs, the Name is a mere identifier for the SubUnit. Class is a simple means of differentiating the role of the SubUnits. The SubUnit Classes are Air, Land, and Sea. These correspond directly with the Unit Classes. When building a Sea Class Unit, one will have a choice of Sea and Air SubUnits. Land Units are constructed from Land and Air SubUnits. Air Units are built strictly from Air SubUnits.

After one enters the SubUnit Name and Class, the list of all Commodities will be displayed. The user will be prompted to choose from the list by entering the Name of the Commodity. If the user enters a valid Commodity Name, the user will be further queried as to the stocking objective of the Commodity for that particular SubUnit. Then the high, medium, and low consumption rates are entered. This process of adding Commodities is repeated until the user is satisfied that the SubUnit is complete. At that point, the user is asked whether the user wishes to modify the SubUnit, or not. If so, the user will be afforded the opportunity to add or subtract a Commodity or modify any of the quantities that have already been specified.

2. Modeling

As previously mentioned, the SubUnit is the basis for building Units. SubUnits make Unit creation as flexible as possible. For example, the CTF 50 Carrier Battle Group would nominally consist of SubUnits that represent ships and aircraft. In this case, the Unit is constructed by aggregating the Commodity requirements of a CVN, its airwing (defined by individual aircraft types), two CGs, a DDG, a DD, and two FFGs. If one wishes to create a different battle group with a different composition, one would not need to recalculate the consumption requirements based on the new organization of ships and aircraft. One would merely build a battle group with a different combination of SubUnits. Given the recent paradigm of joint littoral warfare fought by forces tailored to meet the specific needs of the contingency, the flexibility of the SubUnits system is a sound idea. All that is required of the user is to build a library of commonly used SubUnits.

D. UNITS

1. Construction

The next logical object to discuss is the Unit. As mentioned in the previous section, Units are built by aggregating the Inventory information contained in the SubUnit. But, this is not all that goes into the construction of a Unit. Besides the usual Name field, the Unit has the following other characteristics that must be defined during the construction process:

- Activation Information
- Closure Status
- Class
- Position
- Port Information
- Assigned Transporters
- Inventory.

Activation information concerns the status of the Unit when the scenario begins. A Unit could begin a scenario in several different states defined by its combat intensity, closure status, and activation delay. As previously discussed, a Unit has four combat intensity levels: High, Medium, Low, and None. The Unit may begin the scenario at any of these four levels to fit the scenario conditions. Additionally, if deployment is to be simulated, the Unit may begin in any state of closure. This state is defined by the amounts of equipment already in its inventory or at its origin. This deployment status may be further delayed by entering an activation delay, which is defined by the number of days from the scenario start until the Unit begins the deployment process. This feature has been provided to enable the user to phase unit deployment much in the same way a TPFDD is executed.

The next item is the Unit Class. As mentioned, Unit Class mirrors that of SubUnit's Air, Sea, and Land. This Class structure is defined primarily to determine the eligible SubUnits from which to build the Unit.

Position is a critical element in the creation of a Unit. It is entered by the user when prompted as the latitude and longitude of the intended final location of the Unit when it is deployed. Position is important for determining eligible suppliers and travel distances.

Like Bases, there are four Ports that may be defined for each Unit. Again, these Ports are a seaport, an airport, a rail yard, and a truck stop. To maintain maximum flexibility and a measure of generality in the definition of Units, all Ports are eligible for construction. Naturally, some Port types will not make sense given the nature of the

Unit Class. The user decides which Ports will be present at each Unit. For example, a naval task force would not require a truck stop, but a Marine Expeditionary Unit located at a captured enemy port may conceivably have all four types of Ports available for use. The user must decide which Ports will be appropriate for each Unit in a scenario. Once the Ports have been selected, the user is prompted for the characteristics of the Ports present at the Unit. A Port has two main characteristics that must be defined: capacity, and Transporter maximum size. Capacity is an integer value representing the number of Transporters, which may be unloaded simultaneously. Essentially, this is the maximum size of the Port Berth queue. Transporter maximum size corresponds to Transporter size in that it is intended to be a limiting characteristic. Currently, this characteristic is unused, but is provided for further refinements to the program.

Assignment of Transporters to a Unit is accomplished much in the same manner as selecting Commodities for SubUnits. The user is shown a list of previously constructed Transporters and asked to enter the Names and quantities of selected assets. When selected, the Transporters are created by S3 and placed in the Parked queue of the appropriate Port. This is the primary method for positioning Transporters at Unit locations when building a scenario.

Inventory selection is accomplished using SubUnits. A list of all eligible SubUnits (by Class) is displayed for the user. The user selects a SubUnit by entering the SubUnit name. The user is then prompted for the number of the selected SubUnit that will be assigned to the Unit. When a SubUnit is selected, the items and required quantities of its Inventory are added to the Inventory of the Unit under construction. As a default, the on-hand amounts for each Commodity are set at the SubUnit Stocking Objective for that Commodity. The user will be afforded the opportunity to change these quantities to match scenario conditions later in the Unit construction process. The consumption rates of the Commodities in the SubUnits Inventory are also added to the total Unit consumption rates for that Commodity. This process is repeated until the user is satisfied that all SubUnits have been selected in the proper quantities. At this point, the user is given the opportunity to select additional Commodities not automatically provided by the SubUnit selection process. This concludes the Unit construction procedure.

Upon completion of the construction procedure, the user is granted the opportunity to modify the newly built Unit to catch any errors or to revise its characteristics. This process allows for the modification of any of the Unit characteristics including Transporter assignments and Inventory status. The modification procedure is the same procedure used during simulation execution except that the modifications represent permanent changes to the Unit. During run time, changes to a Unit are only valid during the scenario execution and are not saved after the game. The appropriate way to permanently change characteristics is through the Scenario Editor after Unit creation or by selecting the Modify Unit menu choice after the unit has been built.

2. Modeling

Much flexibility has been built into the Unit construction process. This flexibility allows the user to model particular Units as desired. By using SubUnits, any echelon of Unit may be constructed. The user should attempt to build typical SubUnits to represent the lower echelons of the various types of units in a given class. Ground forces represent this flexibility to the greatest degree. For example, if the user wishes to model ground forces at the corps level, the appropriate SubUnits would be created at the division level. Special units such as combat engineers, reconnaissance platoons and artillery brigades could also be created to tailor the corps level unit further. If user wishes the user may model battalion level units, instead. Then, the appropriate SubUnits would be companies and individual platoons with individual vehicles and munitions as Commodities in their Inventory.

Port selection is a major factor in the proper modeling of scenario Units. At this point in S3's development, there is no limitation to the size of Transporters that may enter a given Port. Therefore, it would be appropriate to model naval task groups as Units that have only a seaport for supply. The user must exercise some creativity to realize that UNREP of a battle group is equivalent to the battle group having limited sea port facilities. This allows shuttle ships to arrive at the Unit to simulate CONREP with the battle group station ships. Of course, ships do not have facilities for handling truck convoys or freight trains so the omission of rail yards and truck stops from Sea Class units would be appropriate. An airport would also be omitted because it would be inappropriate for strategic lift aircraft to bring supplies to forces afloat (COD is assumed to be a minor contribution).

Naturally, Ground and land-based air units would be supplied by rail or truck, but it is feasible that some units, particularly those in amphibious operations, could be supplied directly by sea. Additionally, air units could be directly supported by aircraft or could receive ground-based shipments from a nearby base. Flexibility is provided for the user to determine what method of surge and supply is appropriate for each Unit by selecting the proper Ports during the Unit construction process.

E. BASES

1. Construction

The construction of Bases is very similar to the construction of Units with some exceptions. First, Bases do not use SubUnits for their construction since they have no inherent consumption planning factors. Second, Bases do not have closure and activation information or the same Class structure. Third, Bases are assigned to Groups and SubGroups in during construction, which is done automatically for Units. In all other respects, Bases are almost identical in construction to Units. During construction, the following information is entered by the user:

- Name
- Group
- SubGroup
- Position
- Port Information
- Assigned Transporters
- Inventory.

Like the constructs previously discussed, the Base Name is an identifier unique to each individual Base. Preferably, this identifier will be 15 characters or less in length.

The Group and SubGroup of a Base are assigned by the user during the construction process. Group selections are CONUS, Intermediate Location (ILOC), Theater, or Unit. The Unit Group selection is reserved for the construction of Units <u>only</u>, and is set by the program automatically, during Unit construction. SubGroup selections are Port of Embarkation (POE), Port of Debarkation (POD), Port of Supply (POS) or None. Bases in the None SubGroup do not have the facilities to classify them in another category. These Bases are still eligible as suppliers, but, Shipments will be routed to POEs before transportation to the theater during surge movement.

If a Base is in the None SubGroup, and, in the Theater or ILOC Group, its shipments will be routed normally.

Entry of Base Position is identical to the entry of Unit Position. Transporter assignment and entry of Port information are also identical to that of Units.

The selection of Commodities for Bases, however, is not identical to the selection of Commodities for Units. Since Bases do not have SubUnits, this portion of the Commodity selection process is omitted. The user must stock the Base Inventory Commodity by Commodity. Additionally, consumption rates for Commodities in Base Inventories are not set by the Scenario Editor.

2. Modeling

For the most part, modeling logistics sites with S3 is fairly intuitive. Any geographical location, base, port, unit origin, supply point, etc... may be modeled using the Base construct. The main decision of the user is to determine the scope or scale of the modeled logistics site. For example, if detail is desired, the user could model Bahrain as two bases, one with a sea port and a truck stop, and one with an airport and a truck stop. Here, transportation between the sea port and the airport would be simulated by the movement of Shipments by truck. Alternatively, the user could model Bahrain as a single base with a sea port and an airport without the truck stop. In this case, transportation of Shipments between Bahrain's ports would not be modeled. All Commodities would be "shared" by the sea port and airport since they would originate and arrive at the same Inventory. The selection of detail is entirely a matter of user preference.

When creating bases, care should be taken to ensure that they have the necessary Ports to allow communication with their intended customers and suppliers. For example, if Riyadh is mean to be the POS for Units operating in the Saudi interior, the Riyadh Base must have the necessary over-land transportation facilities. Otherwise, another Base will be selected that does communicate with the intended Units.

In the worst case, S3 will halt for a transportation infeasibility. This will happen if there is no possible route from an eligible supplier to a customer during execution. When this occurs, the user will be forced to enter the Scenario Editor to fix the problem. The user must examine the logistics network to determine where the infeasibility exists. Typically, this infeasibility exists when there is no communication between CONUS Ports of Embarkation and theater Ports of Supply or Debarkation. This occurs if there is no corresponding Port of Embarkation for a Port of Debarkation or Supply. In other words, if the only port in theater is a <u>seaport</u>, a CONUS <u>seaport</u> must also be modeled. The same holds true for airports, as well. Given the nature of the logistics problems and the availability of sea and air transportation from CONUS, it is unlikely that a user would neglect to ensure adequate port communication.

F. PREPOSITIONED TRANSPORTERS

Prepositioned Transporters (Prepo) have been created primarily to allow the modeling of the various prepositioned forces available to a joint commander in the event of a contingency. These would include the Maritime Prepositioned Force (MPF), the Afloat Prepositioned Force (APF), the new Army afloat prepositioned forces, and to a limited extent, the Fast Sealift Ships (FSS). Since S3 is written to take advantage of the generality of object-oriented programming, it is possible to define Prepo assets for more than just these limited choices. Any Transporter may be prepositioned with its Cargo in any location desired by the user. This affords maximum flexibility in the design of a scenario.

1. Construction

Prepo assets, of themselves, are not special constructs in the S3 program. They are merely previously constructed Transporters that start a scenario in a specified location (other than a Base or Unit) and have Cargo designated for a particular Unit. The process of constructing a Prepo asset for a scenario involves the selection of a Transporter, the specification of its Position and the selection of its Cargo and destination.

To begin the Prepo construction process, the user selects from a displayed list of all previously constructed

Transporters. Thus, the user must have already built the Transporter that the user wishes to use in the prepositioned role before the creation of a Prepo asset. The user selects the Transporter by entering its Name when prompted by S3.

Once the user selects a valid Transporter, the user is asked to provide the location of the Transporter in latitude and longitude. The user is then asked to select the name of a Base or Unit from a list of eligible Bases and Units. This Base or Unit represents the ultimate destination of the Transporter Cargo. Then the user iteratively builds the Cargo routing by selecting intermediate destinations from the list of Bases. The first intermediate destination selected will be the Transporter's assigned destination to which it will deliver the Cargo. After the Transporter arrives at this Base, the Cargo will continue to its ultimate destination. Special care must be taken to ensure that the Bases or Units selected as destinations for the Transporter and Cargo are modeled in the scenario in question. If they are not, a run time error will occur, as the Base selected will not exist within the scenario.

After entering the routing information, the specifics of cargo are defined. The user selects the Commodities that comprise the Transporters Cargo in the same way that the user would build the Inventory of a Base. It is important to note that the normal limitations of weight, cube and area are not in effect during the Prepo construction process. Therefore, care should be taken not to overload the transporter. This will not result in any harm to the scenario or program, although, it does allow unrealistic Cargo configurations to be constructed. Once the user is satisfied with the Cargo of the Transporter, the process of constructing the Prepo is complete and the asset is saved to a data file.

2. Modeling

Modeling the prepositioned forces in a scenario is simple and straight forward. As long as the Transporter and the Commodities in its Cargo have been previously constructed, there should be no difficulty in creating a Prepo. As previously mentioned, the Prepo can be used to model any cargo vessel that is enroute at the beginning of a scenario. This includes aircraft, trucks, and trains, if needed. Typically, this feature will be used to model APF and MPF as well as the occasional shuttle ship that is already enroute to a battle group.

G. SCENARIOS

Construction of the scenario is fairly simple, once the preliminary work is finished. When the user builds a scenario, the user will be asked to provide the Scenario Name. Like other Names, the Scenario Name should be limited to fifteen characters as a matter of convention.

The user will then be asked to pick the Bases that will be active during the game. At this point, the user will be able to view a list of all constructed Bases, view the list of selected Bases, add a new Base to the selection list, delete a Base from the selection list, or continue to the next step of the scenario building process. Once the user is satisfied with the base selections, the user may continue the scenario construction procedure.

The user's next step is to select the Units which will be available during the game. This Unit selection process is identical to the previous base selection process except that only Units will be listed. The user is then asked to select the Prepositioned Transporters that will be active during the game. Again, this process is similar to the Base selection process.

After Prepo selection is accomplished, the user will link the transportation networks of the Units and Bases. First, the rail networks will be linked. In this procedure, the user will be presented with a Base or Unit. The user should select all Bases that communicate by railroad with the presented Base or Unit. Each Base and Unit will be presented in turn. Once the rail network is entered, the road network will be constructed. This process is identical to the rail network construction process. When the user has finished construction of the road network, the Scenario is complete. A scenario file will be generated, and the scenario will be added to a master list from the user will select the scenario to execute.

V. USING THE MODEL

An effective evaluation of the Surge and Sustainment Simulation requires the demonstration of its capabilities in an actual wargame scenario. For the purposes of this demonstration, the scenario will be the Surface Warfare Officer School Game 129 (SWOS-129). This game was conducted at the Naval War College, War Games Department on 2-3 NOV 1993 on the ENWGS system. Scenario information was provided by the Logistics Branch of the War Game Department, the intended user of the S3 model. Only the scenario initial setup was utilized; the actual game events were not available at the time of this writing. All game events were created specifically to show the capabilities of the model in a wargame environment.

A. THE SCENARIO

1. Background

Blue forces must establish local naval and air superiority to display commitment to freedom of the seas and the territorial integrity of Green, an Islamic parliamentary democracy located near the Gulf of Oman, on the Southeast Asian coast. Additionally, Blue forces must be prepared to carry out Non-Combatant Evacuation (NEO) operations of Blue civilians from the Green port of Pintoint, and, must be prepared to carry out offensive operations in support of

Green's defense of its territorial integrity against Orange. The vital goal of the operation is to ensure that the port of Pintoint remains open and viable.

Orange is an Asiatic neighbor of Green to the south and west. A former client state of Red in the old bipolar geopolitical order, Orange has maintained tense, even hostile, relations with Green since Green's establishment earlier in the century. This tension has not been eased by the secessionist movement of the predominately Muslim population in fertile Northwestern Orange. The Orange government is convinced of Green's complicity in the movement and is prepared to back its diplomatic warnings with force if necessary. Of course Blue, having maintained friendly and cooperative security relations with Green, cannot allow Orange to infringe on the territorial sovereignty of Green. Nevertheless, Blue does not wish to provoke Orange to open conflict with Green, but will take steps to insure the success of Blue forces should a conflict arise.

To that end, the following Blue Forces are provided to the JTF Commander:

- Naval Forces

COMMANDER, MID-EAST FORCE

USS LaSalle (AGF-3)
USS Stout (DDG-55)
USS Jarret (FFG-33)
USS Samuel B. Roberts (FFG-58)
USS Avenger (MCM-1)
USS Patriot (MCM-7)

COMMANDER, CARRIER STRIKE FORCE 50 (CTF-50)

USS Abraham Lincoln (CVN-72)
- COMCARGRU ONE
- COMDESRON TWENTY THREE
- COMCARWING ONE
  - 24 F-14D
  - 24 F/A-18C
  - 12 A-6E
  - 4 EA-6B
  - 5 E-2C
  - 8 S-3B
  - 2 ES-3
  - 6 SH-60F
  - 2 HH-60H

USS Chancellorsville (CG-62)
USS Shiloh (CG-67)
USS John S. McCain (DDG-56)
USS O'Brien (DD-963)
USS Rentz (FFG-46)
USS Vandergrift (FFG-48)

COMMANDER, AMPHIBIOUS TASK FORCE 51 (CTF-51)

TG 51.1 Amphibious Task Force

Amphibious Ready Group Alpha

USS Essex (LHD-2)
- 6 AV-8B
- 12 CH-46E
- 4 CH-53E
- 3 UH-1N
- 4 AH-1W
- 3 LCAC

USS Germantown (LSD-42)
- 3 LCU

USS Harpers Ferry (LSD-49)
- 2 LCAC

WEST PAC MEU SOC
- 6 155MM Howitzers
- 2 105MM Howitzers
- 1 LAV-C2
- 1 LAV-L
- 2 LAV-M (81MM Mortar)
- 7 LAV-25

2239 Personnel Embarked

Amphibious Ready Group Bravo

USS Saipan (LHA-2)
6 AV-8B
12 CH-46E
4 CH-53E
3 UH-1N
4 AH-1W
4 LCU
USS Whidby Island (LSD-41)
4 LCAC
USS Carter Hall (LSD-50)
2 LCAC
MED MEU SOC
6 155MM Howitzers
2 105MM Howitzers
1 LAV-C2
1 LAV-L
2 LAV-M (81MM Mortar)
7 LAV-25
2274 Personnel Embarked

TG 51.2 Evacuation Group

USS John Young (DD-973)
USS McCluskey (FFG-41)
MV Kepwave
MV New Venture
MV Cygnus
MV Dunedin
MV Rostand

COMMANDER, SUBMARINE TASK FORCE 52 (CTF-52)

USS Helena (SSN-725)
USS Chicago (SSN-721)

COMMANDER, MARITIME PATROL & RECONNAISSANCE FORCE 53 (CTF-53)

VP-1	(8 P-3C)	Diego Garcia
VP-8	(8 P-3C)	Masirah
VQ-1	(2 EP-3)	Masirah

- Air Force

Masirah, Oman
6 F-15C
12 F-15E

Riyadh
5 E-3B (AWACS)
6 KC-135R

- Army Forces

82nd Airborne Division, Ft Bragg, NC
3rd Armored Cavalry Regiment, Ft Bliss, TX

2. Scenario Setup

a. *Commodities*

The SWOS-129 scenario indicates that the following commodities are of probable interest to the players. For the purposes of the example these Commodities will be created using the S3 Scenario Editor:

- Commodity Class: Fuel

JP-5	DFM	Water
DIESEL	MOGAS	

- Commodity Class: FFV

FFV	Frozen Goods	Dry Stores

- Commodity Class: Ammo

Missiles:

TLAM-N	TLAM-C	TLAM-D
TASM	Harpoon (Air)	Harpoon (Sea)
ASROC	Sea Sparrow	SM-1MR
SM-2MR	SM-2ER	HARM
AIM-54C	AIM-9M	AIM-9L
AIM-7M	AMRAAM	AGM-65
AGM-62	Penguin	TOW II
HELLFIRE		

Guns:

20MM	20MM/76	25MM
40MM Grenade	76MM/62	76MM/50
81MM Mortar	105MM	127MM/54
155MM		

Bombs:

Mk-80 Series	Paveway II	APAM
Rockeye	FAE	

Misc:

Sonobuoys	Mk-46 Torpedo	Mk-48 ADCAP
Mk-54 Depth Charge	SRBOC	

- Commodity Class: Major

M-1A1	AH-64	M-109
105HOW	UNITEQ	

- Commodity Class: Other

Mail

For simplicity, all missiles, bombs, and miscellaneous Ammo Class Commodities, have been represented as individual rounds, while gun ammunition, rations, and miscellaneous Unit Equipment have been represented as pallet shipments of one or two Short Tons (STONS). All liquid Commodities are represented as 42 gallon barrels (bbls). Of course, all major end items (except UNITEQ) are represented as individual pieces of equipment. Appendix B lists the gives Commodity data files for the SWOS-129 scenario.

b. *Transporters*

The following Transporters have been built for the SWOS-129 scenario:

- C-5A Galaxy aircraft
- C-141B Starlifter aircraft
- B-747 CRAF aircraft
- Tanker (200,000 bbls.)
- Comet class RoRo cargo ship
- SL-7 class RoRo cargo ship
- C-4-S-1H breakbulk cargo ship
- Generic breakbulk cargo ship
- AE26 class ammunition ship

- AO177 class oiler
- Tank Truck Convoy (10 5000 Gal. vehicles)
- Breakbulk Truck Convoy (10 5-ton vehicles)
- Unit Equipment Train (50 54' flatcars)
- Freight Train (40 59'9" boxcars)
- Passenger Train (7 coaches, 14 sleepers, 1 baggage car)
- Tank Train (40 20,000 Gal. tank cars)

Appendix C provides the Transporter data files for the SWOS-129 scenario.

c. *SubUnits*

Based on scenario requirements, the following SubUnits must be created through the model Scenario Editor:

- Ships:

CVN68	CG47	CG52
DDG51	DD963	FFG7
LHD1	LSD41	MCM1
AE27	AFS1	AO177
AOE1	TAO187	LCAC

- Aircraft:

F-14D	F/A-18C	A-6E
EA-6B	E-2C	S-3B
ES-3	SH-60B	SH-60F
HH-60H	P-3C	EP-3
KC-135	E-3B	F-15C
F-15E	AV-8B	UH-1N
CH-46E	CH-53E	AH-1W
AH-64		

- Ground:

155MM Howitzer	105MM Howitzer	LAV-C2
LAV-L	LAV-M	LAV-25
M-1A1	M-109	

Appendix D provides the SubUnit data files for the SWOS-129 scenario.

d. *Units*

For the SWOS-129 example, the following Units have been constructed from the indicated SubUnits:

- Middle East Force (MEF)
  1 AGF3
  1 DDG51
  2 FFG7
  2 MCM1
  2 SH-60B

- COMMANDER, TASK FORCE 50 (CTF50)
  1 CVN68
  2 CG52
  1 DDG51
  1 DD963
  2 FFG7
  24 F-14D
  24 F/A-18C
  12 A-6E
  4 EA-6B
  5 E-2C
  8 S-3B
  2 ES-3
  8 SH-60F
  4 SH-60B

- TASK GROUP 51.1 ALPHA (TG51.1A)
  1 LHD
  2 LSD41
  6 AV-8B
  12 CH-46E
  4 CH-53E
  3 UH-1N
  4 AH-1W
  3 LCAC

- TASK GROUP 51.1 BRAVO (TG51.1B)
  1 LHD
  2 LSD41
  6 AV-8B
  12 CH-46E
  4 CH-53E
  3 UH-1N
  4 AH-1W
  4 LCU

- TASK GROUP 51.2 (TG52)
  1 DD963
  1 FFG7

- Air Force, Riyadh (AF-RIYADH)
  5 E-3B
  6 KC-135R

- Air Force, Masirah (AF-MASIRAH)
  6 F-15C
  12 F-15E

- VP-8, Masirah (VP-8)
  8 P-3C

- VQ-1, Masirah (VP-8)
  2 EP-3

- VP-1, Diego Garcia (VP-1)
  8 P-3C

- Seventh Marine Expeditionary Brigade (7MEB)
  12 M-109
  4 105HOW
  2 LAVC2
  4 LAVL
  4 LAVM
  14 LAV25

- 3rd Armored Cavalry Regiment (3ACR)
  116 M-1A1
  18 M-109
  9 AH-64

- 82nd Airborne Division (82ABNDIV)
  33 AH-64
  54 105HOW

Appendix E provides the Unit data files for the SWOS-129 scenario.

e. *Bases*

For the SWOS-129 scenario, the following Bases have been modeled:

- Theater Bases

  Masirah, Oman: Airhead, no stocked Commodities
  Pintoint, Green: Port of Debarkation

- Intermediate Locations (ILOC)

  Djibouti: Fuel stock point
  Bahrain: Fuel stock point
  Muscat, Oman: Fuel stock point
  Diego Garcia, BIOT: Fuel and ordnance stock point
  Sigonella, Italy: Fuel stock point
  Rota, Spain: Fuel stock point
  Singapore: Fuel stock point
  Okinawa, Japan: Fuel stock point
  Guam: Fuel and ordnance stock point

- U.S. Bases (CONUS)

  NWS Earle, NJ: Ordnance stock point
  NWS Concord, CA: Ordnance stock point
  MOT Bayonne, NJ: Port of Embarkation
  MOT Oakland, CA: Port of Embarkation
  NSC San Diego, CA: Fuel and ordnance stock point
  NSC Norfolk, VA: Port of Embarkation
  Ft Bragg, NC: Origin, 82nd Airborne Division
  Pope AFB, NC: Air Port of Embarkation
  Wilmington, NC: Sea Port of Embarkation
  Ft Bliss, TX: Origin, 3rd Armored Cavalry Division
  Holloman AFB, NM: Air Port of Embarkation
  Corpus Christi, TX: Sea Port of Embarkation

Appendix F provides the Base data files for the SWOS-129 scenario.

f. Prepositioned Transporters

For the SWOS-129 scenario, the following Prepos have been created and positioned.

- MPS Squadron 2

  Bonneyman: Diego Garcia, BIOT

- Fast Sealift Ships

  Capella: Jacksonville, FL
  Altair: Norfolk, VA
  Regulus: Violet, LA
  Pollux: Violet, LA
  Bellatrix: Galveston, TX
  Algol: Galveston, TX
  Denebola: Bayonne, NJ
  Antares: Jacksonville, FL

Appendix G provides the Prepo data files for the SWOS-129 scenario.

B. THE GAME

After the scenario files have been built, S3 is ready to begin the logistics simulation for the wargame. The Logistics Umpire will maintain communication with the umpires on the "game floor" and the players through an assigned facilitator. This communication is essential to the effectiveness of the S3 model. As there is not a direct link between the wargame and the logistics model, the Logistics Umpire must ensure that important game events are reflected in inputs to S3. Additionally, the Logistics Umpire must effectively communicate the logistical status of forces in the wargame to the players and their facilitators.

1. Initial Response

The game opens with the elements of the Blue naval forces already on scene. At this point, the surging units, the 82nd Airborne Division and the 3rd Armored Cavalry Regiment, are at their origins, awaiting orders to deploy. These units will be held until the crisis warrants the deployment of ground forces to the Pintoint region of Green. Meanwhile, the Blue naval units will be conducting operations designed to show a determined presence in the region without resort to actual hostilities. As a result, all Units in the theater should be set initially to the None Combat Intensity Level. At this setting, the Sea and Air Class Units already deployed the theater will be expending mainly fuel and limited ordnance Commodities, such as sonobuoys. Any expenditure of other ordnance at this point, would be handled on a case by case basis. This represents the effect of the minor, yet provocative, engagements that occur as a crisis develops to full proportions.

S3 responds to the game situation by automatically scheduling the resupply of the naval task forces and the land-based air components. Every 24.0 simulated hours, each Unit will consume the Commodities in its Inventory according to the "None" Combat Intensity. When a Commodity OnHand level reaches its Order Point, the Unit will place an order for that Commodity with the Logisitics Manager. The Logisitics Manager will determine the best supplier for the Commodity and pass the request on to the appropriate base. Literally thousands of requests will be filled by the end of S3's execution. All of these requests will be handled by S3 automatically, and will be completely transparent to the user.

Currently, all naval ordnance Commodities are replenished from Diego Garcia, the nearest stock point. Requests for fuel Commodities, F44 and F76, are filled by the closest stock point at Muscat, Oman. When Muscat's fuel stocks are reduced to the order point by the Transporters shuttling back and forth to the task forces, the Base will order more from the nearest eligible CONUS supplier, in this case, NSC Norfolk. If Muscat runs out of fuel, the Logistics

Manager will fill Unit requests for fuel from an alternate location, the closest Base to the ordering Unit. If Muscat has an independent fuel source, this may be simulated by the Logistics Umpire periodically adding fuel to the Muscat Inventory by direct modification of its Inventory. Otherwise, all Commodities at ILOC Bases are automatically resupplied from the nearest CONUS Base with the Commodity in stock.

2. First Blood

On the second day of the wargame, two F/A-18 Hornets are fired upon by a flight of Orange Mig-23s. In the brief but intense air battle, two Mig-23s are shot down by AIM-9M "Sidewinder" air-to-air missiles. A total of four missiles were expended by the Hornet flight.

The Logistics Umpire responds by reducing the On-Hand levels for AIM-9M in the CTF50 Inventory by four missiles. The current order point for Commodity AIM-9M is 135 missiles. With the loss of these four, the On-Hand level of AIM-9M for CTF50 is now 146. Since the On-Hand level has not reached the Order Point, no action will be taken automatically by S3. If, through further combat, the level of AIM-9M drops to 135, an order for more missiles will be placed by the Unit.

3. The Fight Begins

Using the shoot-down of its fighters as justification for hostilities, Orange launches a ground offensive against Green on the fourth day of the crisis. The Prime Minister of Orange vows to "sweep Green and her U.S. lackeys into the

Indian Ocean." That day, a large scale raid attempts a strike at the U.S. naval forces in the Arabian Sea. Although none of the U.S. ships are hit, the raid results in the expenditure of 50 SM-2MR missiles by the Carrier Battle Group (CTF50). Responding, the National Command Authority grants the use of force against all threatening aircraft and vessels approaching the battle group. NCA further orders the Amphibious Task Groups to land their Marines ashore at Pintoint to cover the evacuation of U.S. non-combatants. The Green government applauds the U.S. commitment and makes public its desire for more reinforcements from the United States.

The raid by Orange forces against CTF50 is handled in the same way as the previous air-to-air engagement. The 50 SM-2MR missiles are subtracted from the CTF50 Inventory. The CTF50 Unit stocks 304 SM-2MR missiles in its Inventory and orders more when the stock gets below 276. The loss of the 50 missiles results in a request for replacements.

The Logistics Manager then determines that the closest eligible supply of SM-2MR is the ordnance stock point at Diego Garcia. Since the Emergency Order Point for the SM-2MR missile has been set at 90 percent of the total CTF50 Stocking Objective, the SAM shipment is bumped to Priority one. In this case, the missiles will be routed from Diego Garcia, an Intermediate Location, to Masirah, a Theater base. From there, the shipment will travel by sea to the battle group. The Logisitics Manager builds a Shipment, including the desired item and route, and passes it to the Diego Garcia Base.

The Base responds by reducing its Inventory and adding the 50 missiles to the Shipment, which is then sent to Diego Garcia's airport. The Port places the Shipment in a queue of Shipments bound for Masirah, the next stop on the route. Since the missiles are the first Shipment travelling by air to Masirah from Diego Garcia, the Port requests a aircraft through the Transporter Manager.

The Transporter Manager searches through its queue of available aircraft and determines that there is a C-5 ready at Diego Garcia. The Transporter Manager orders the C-5 to make itself available to the Diego Garcia Airport. After loading the 50 missiles and any other Shipments which might have arrived for Masirah, the C-5 departs for the Base at Masirah.

When the C-5 reaches Masirah, it will automatically unload its cargo. Since Masisrah does not maintain any Commodities in Inventory, all Shipments will be passed to the appropriate Port to continue their journey. Otherwise, the Shipments would have been sorted to determine if any belonged to the Base. The 50 SM-2MR missiles are routed the Masirah Seaport, where the process of procuring and loading a Transporter is repeated. The missiles will arrive at CTF50 by Day Six, without any intervention from the user. Whenever a shortfall in Base or Unit Inventory is discovered by S3, this process is repeated.

After the successful landing of Marines at Pintoint, the Logistics Umpire can activate the 7MEB Unit and turn on the Pintoint Ports. This occurs on Day Five. As a 7MEB is supported by amphibious ships for the first thirty days of its deployment ashore, the Unit reflects this fact by arriving in Pintoint fully stocked with Commodities, except for fuel. The 7MEB Air Combat Element will remain aboard ship during the initial stages of the deployment, so the amphibious Units will continue to maintain their consumption of aviation fuel and ordnance Commodities.

As a result of the opening of Pintoint's Port facilities to Blue forces, the S3 Logistics Manager automatically routes all shipments through Pintoint as this Base is now the closest available Theater Base to active game Units. Up to this point, most shipments where routed through Masirah, Oman, depending on the location of receiving Units. Shipments arriving from ILOC or CONUS Bases will now go directly to Pintoint.

4. Surge

On Day Five, the NCA decides to send the 82nd Airborne Division and the 3rd Armored Cavalry Regiment to support Green forces on the Orange frontier. Immediately, these units move to their mobilization stations and begin deployment to the theater. All of the resources of TRANSCOM have been made available to deploy these units to Pintoint. Additionally, eight Fast Sealift Ships (FSS) on the East Coast are activated and proceed to Wilmington, SC. and Corpus Christi, TX to meet the deploying unit equipment. The President also authorizes strikes by CTF50 against selected Orange targets and to support defending Green forces on the border.

The Logistics Umpire responds to these game developments by activating the two surging Units and increasing the Combat Intensity of CTF50, CTG51.1A and CTG51.1B to "Medium." Each of these Units will simulate the expenditure of ordnance in strikes and moderate combat against Orange by automatically subtracting daily consumption rates from specific Commodities. Since most threat weapons have no consumption rate even at "Medium" Combat Intensity, they will be expended by direct user intervention as the game develops.

The Logistics Umpire also activates the Prepositioned Transporters representing the FSS and MPS Squadron Two. To simulate the delay in activating the FSS, the Logistics Umpire will activate five ships, Algol, Antares, Bellatrix, Denebola, and Pollux, at Simulation Time 192.0 hours (Day Eight). The remaining three, Altair, Capella, and Regulus will be activated at Simulation Time 264.0 (Day 11). The Bonneyman in Diego Garcia will be activated at time 144.0 (Day Six) to coincide with the commitment of the Marine Expeditionary Brigade.

As the game unfolds, S3 will automatically keep the Units supplied as Commodity and Transporter availability permits. The Logistics Umpire must track the supply status of each Unit in order to relay this information to players, facilitators and other umpires. As Unit Inventories decrease, certain tactical options will no longer be available to the players. This should be reflected in the decisions and action of umpires and the advice of the facilitators to the players.

In addition to scheduling the supply of active Units, S3 will now handle the deployment of ground forces to the theater. The timing of arrivals of Unit equipment in Pintoint and at the Unit destinations is based on the simulated movement of the Commodities that comprise these Units by Transporters from Base to Base. Each Commodity that is identified as important to the deployment of a Unit is moved from the Unit's Origin in CONUS, through the logistics pipeline to the Unit's destination. S3 will automatically handle the deployment as if the Unit had ordered the Commodity during the sustainment phase of an operation. The arrival of a Unit in theater is the result of interaction with other Unit deployment activities and the logic of the S3 model in scheduling and assigning Transporters to certain tasks.

The closure of the surging Units in the theater will be indicated by the arrival of Unit Commodities at their destinations, and by the change in Unit Deployment Status "Closing" to "In Place." This change in status is performed automatically by S3 when a Unit consumes Commodities each simulated day. For this particular scenario, the change in Unit Deployment Status will occur when the 90 percent of the

Unit Deployment Commodities reaches the Unit Inventory. This 90 percent level is set by the Logistics Umpire during the Unit creation process of the Scenario Editor. This level may vary from zero to one as desired by the user, but may be modified during the game.

5. Day Nine and Beyond

By Day Nine, the first of the FSS ships have reached their loading ports and are awaiting the arrival of unit equipment coming by rail from Ft Bliss and Ft Bragg. When this equipment arrives, it will be loaded aboard and shipped to Pintoint.

Meanwhile, personnel are already beginning to arrive at the Third Armored Cavalry Regiment and the 82nd Airborne Division locations in Pintoint. These personnel were transported by rail to their Air Ports of Embarkation at Pope AFB, NC and Holloman AFB, NM. Here they were loaded aboard Civil Reserve Air Fleet (CRAF) 747s for the flight to Green.

It is obvious that it makes little sense to send the entire complement of these ground units to the theater well in advance of their equipment. However, because S3 handles all Commodities on the basis of priority, the logic guiding S3 causes the model to move personnel to the theater as rapidly as possible. Personnel are thus moved to the theater as fast as conditions will allow without regard to the remaining Unit Commodities. As a result, Personnel arrive at the Unit before any equipment.

Table II FSS LOADING AT TIME 288.0

Algol	50 M-1A1 Tanks 4 AH-64 Helicopters 11 STONS Unit Equipment
Antares	33 AH-64 Helicopters 54 105MM Howitzers 8811 STONS Unit Equipment
Bellatrix	5 AH-64 Helicopters 6078 STONS Unit Equipment
Cappella	2126 STONS Unit Equipment
Pollux	66 M-1A1 Tanks 56 STONS Unit Equipment
Regulus	8504 STONS Unit Equipment

By Day Ten, it is possible for the user to see evidence of the movement of the deploying unit equipment. The Commodities that are in the process of moving may be located using the Locate Commodity feature of S3. When the user invokes this feature and attempts to locate the Commodity, UNITEQ (Unit Equipment), the user will see that Ft Bliss and Ft Bragg have already sent all of their respective Units' equipment to the rail yards for loading aboard trains to the nearest Sea Port of Embarkation. Further investigation reveals that much of this equipment has been already loaded aboard the FSS ships. However, a large portion of the remainder is on trains bound for either Corpus Christi, TX or Wilmington, NC. By Day 13, all of the FSS RoRos will be loaded or in the process of loading Commodities bound for the Third Armored Cavalry Regiment and the 82nd Airborne Division. Table II provides a listing of these first shipments of equipment in the deployment phase of the operation.

6. The Ground War

On Day 12 the Seventh Marine Expeditionary Brigade has completed its NEO operations and is free to support Green in its struggle against Orange. The Bonneyman has recently made port and is unloading additional unit equipment and armor at Pintoint. Meanwhile, things are going poorly for both sides in the War for Southeast Green. Because of the gallant efforts of the Army of Green, the Peoples Army of Orange is far behind in its ambitious timetable. But, this good fortune is due to change--the forces of Green are nearing the breaking point. Responding to the appeals of the Green government, the President has authorized defensive action for the Seventh MEB. The Seventh will move forward to the town of Kotri, a key crossing town on the banks of the Indus. By sitting athwart the main road to Pintoint, the Brigade will form the center of the latest Green defensive positions along the Indus. It is hoped that the Marine Brigade can bolster the Green line at this critical point, providing the beleaguered Green forces will needed relief. With the Marines holding the line, the forces of Green eagerly await the arrival of the deploying Blue units to turn the tide.

In order to simulate this turn of events, the Logistics Umpire relocates the 7MEB Unit to the position indicated by the players battle plans. After accomplishing this position change, the Logistics Umpire can increase the Combat Intensity of the Seventh to High, allowing the automatic expenditure of Commodities to reflect the heavy combat in defense of the Indus River Line. The Seventh MEB will only have a few days of supply at this high rate of expenditure. The Logistics Manager must carefully monitor the Unit's supply status in order to advise the players of the situation at the front.

On Day 24, after eleven days of heavy combat, the situation at the Indus front has stabilized sufficiently. Unfortunately, the Seventh Marine Expeditionary Brigade has less than half of its unit equipment left, and will be unable to mount an effective counterattack. The players decide that the Seventh will maintain position, patrol its sector, and build up its supplies while waiting for reinforcement from the 82nd Airborne and Third Armored Cavalry. To simulate this new situation, the Logistics Umpire merely resets the 7MEB Unit Combat Intensity level back to the None Combat Intensity level.

In the meantime, the deployment situation continues to improve. By time 336.0 (Day 14), all of the 4663 personnel of the Third Armored Cavalry Regiment have arrived in Pintoint. At this point, 82nd Airborne Division now has 9325 of its

11674 man complement. By 528.0 (Day 22), both Units have received all of their tanks, helicopters, artillery, fuel, and a fair portion of their miscellaneous equipment. The deployment of miscellaneous unit equipment will continue throughout the game. By 768.0 (Day 32) 27 days after the decision to mobilize, the required 90% of the 82nd Airborne Division has deployed to Green.

The closure of these Units is summarized by Figure 9. The graphic shows that S3 gives a fairly accurate depiction of Unit deployment. The large jumps in percentage illustrate the discrete nature of the arrival of Shipments at Pintoint. It is hoped that later versions of S3 will feature a graphic display like Figure 9.

7. Sustainment

When sufficient Deployment Commodities arrive, Units change their deployment status from "Closing" to "In Place." At this point, they begin to automatically consume Commodities at the default Combat Intensity set during the Unit creation process or at a level set by the Logistics Umpire during the game. S3 will process the replacement of consumed Commodities in the same manner that it has for all the other Unit that were in place at the beginning of the game As usual, the Logistics Umpire must monitor the Inventory levels of the various Units to ensure that the players are up to date on the supply situation. Of course, when combat occurs, the Logistics Umpire may find it necessary to directly affect the

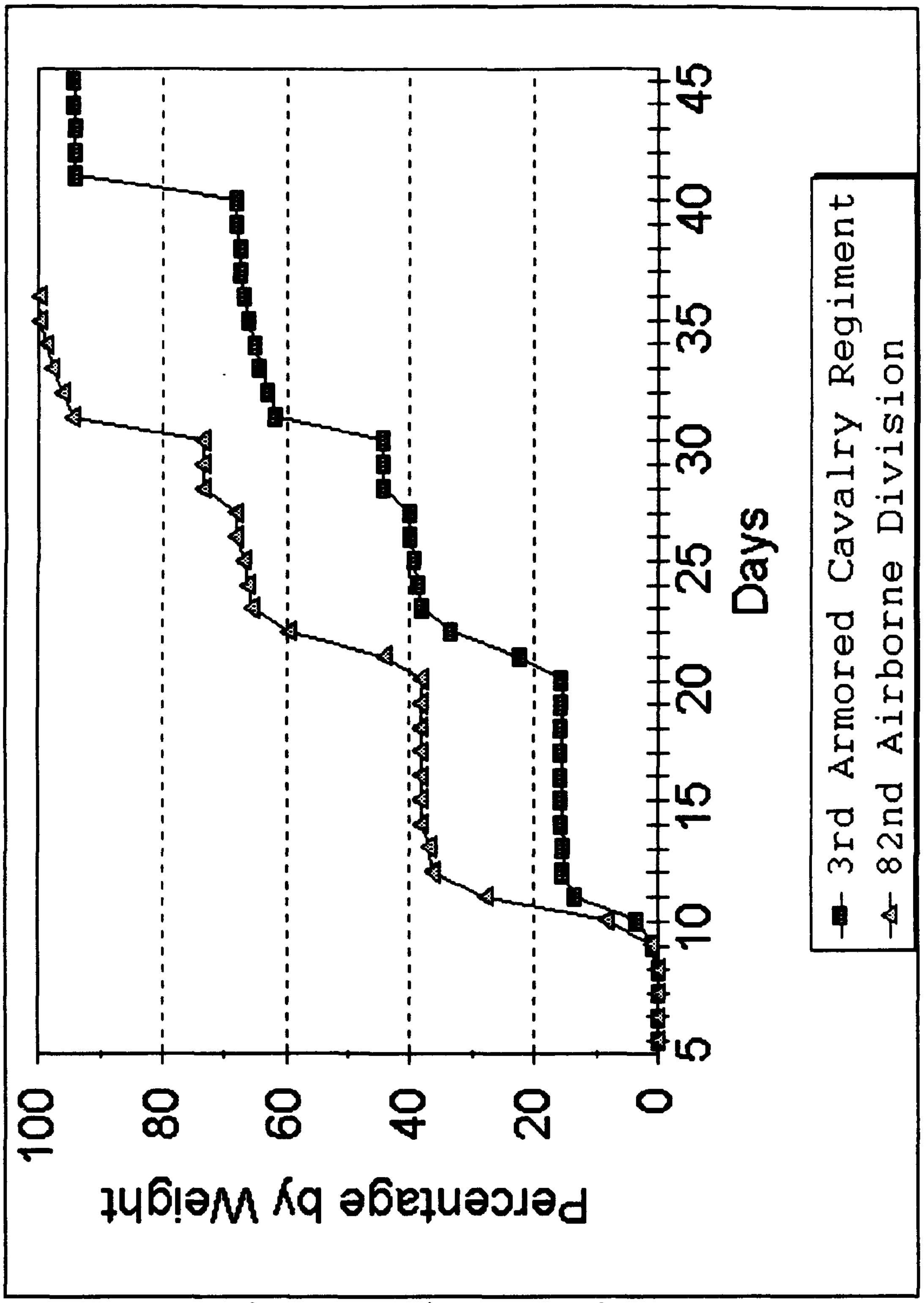


Figure 9 Unit Closure by Day

Inventory of any Unit to reflect events in the game.

Figures 10 through 13 show the supply status of several Commodities in the Inventory of the CTG51.1 Alpha Unit. The steady decline of supply levels as Commodities are consumed followed by the sharp increase as Shipments are delivered is typical of inventory models. The graphs illustrate the success of the S3 in modeling sustainment. It is of importance to note that S3 was able to keep up with Unit demands for Commodities and that no Commodity drops to an unacceptable level.

8. Aftermath

By the time the 82nd Airborne is fully deployed, the government of Orange has realized its mistake. Now facing a renewed effort by the Green-Blue joint command, Orange decides that it would be better to save its strength for later conflicts. Rather than oppose a determined Blue with its reputation for quick success on the ground, Orange begins to pull its invasion force back to its starting positions. As the Prime Minister of Orange stated, "We have made our point, the succession of the State of the Valleys will not be tolerated. There is no need for further bloodshed."

C. SUMMARY

The above scenario was designed to demonstrate the intended use of S3 as a wargaming aid. The creation and execution of the scenario also provided a means to evaluate the effectiveness of the Surge and Sustainment Simulation at

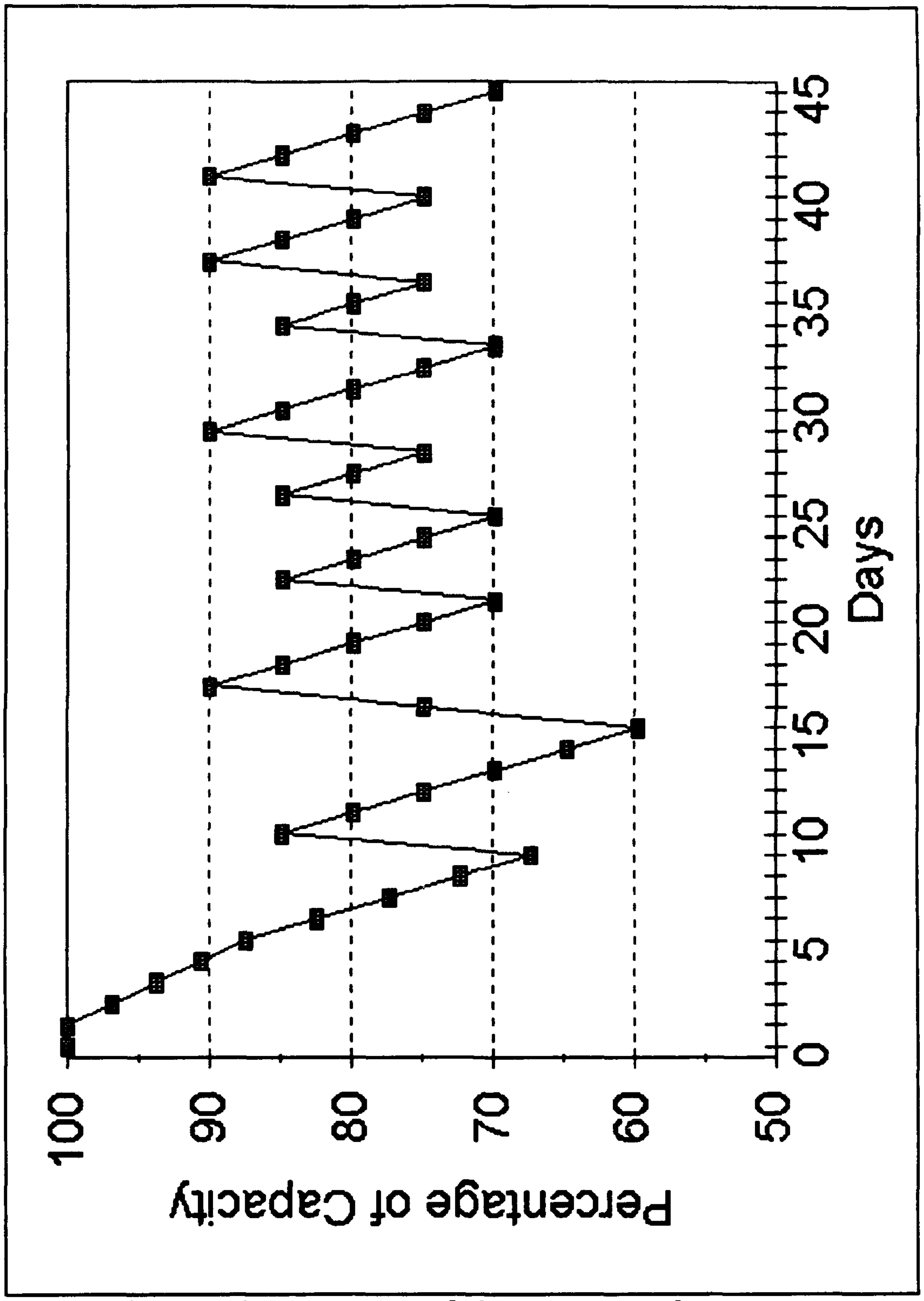


Figure 10 CTG 51.1 Alpha F-44 Supply Status

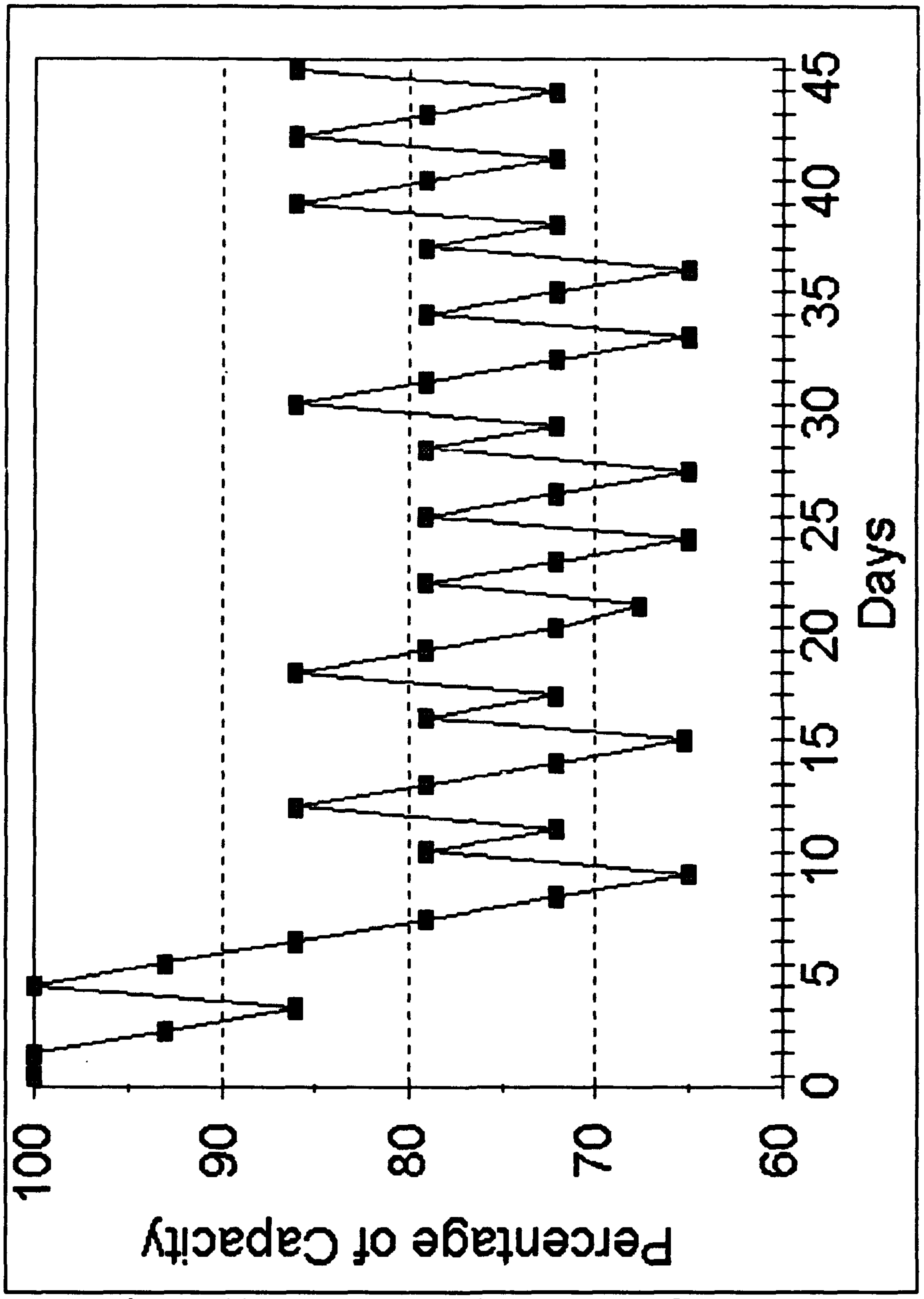


Figure 11 CTG 51.1 Alpha F-76 Supply Status

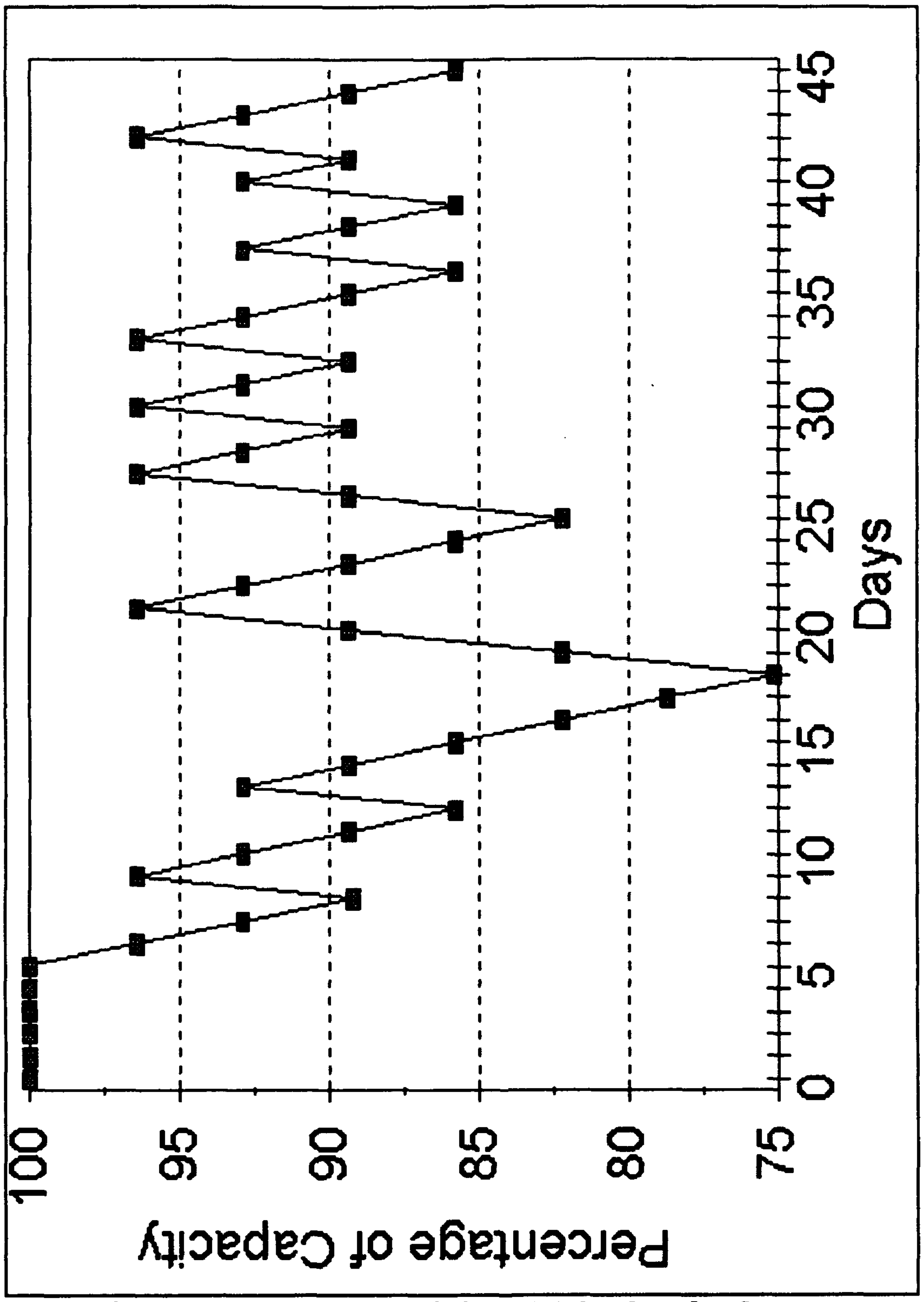


Figure 12 CTG 51.1 Alpha AIM-9L Supply Status

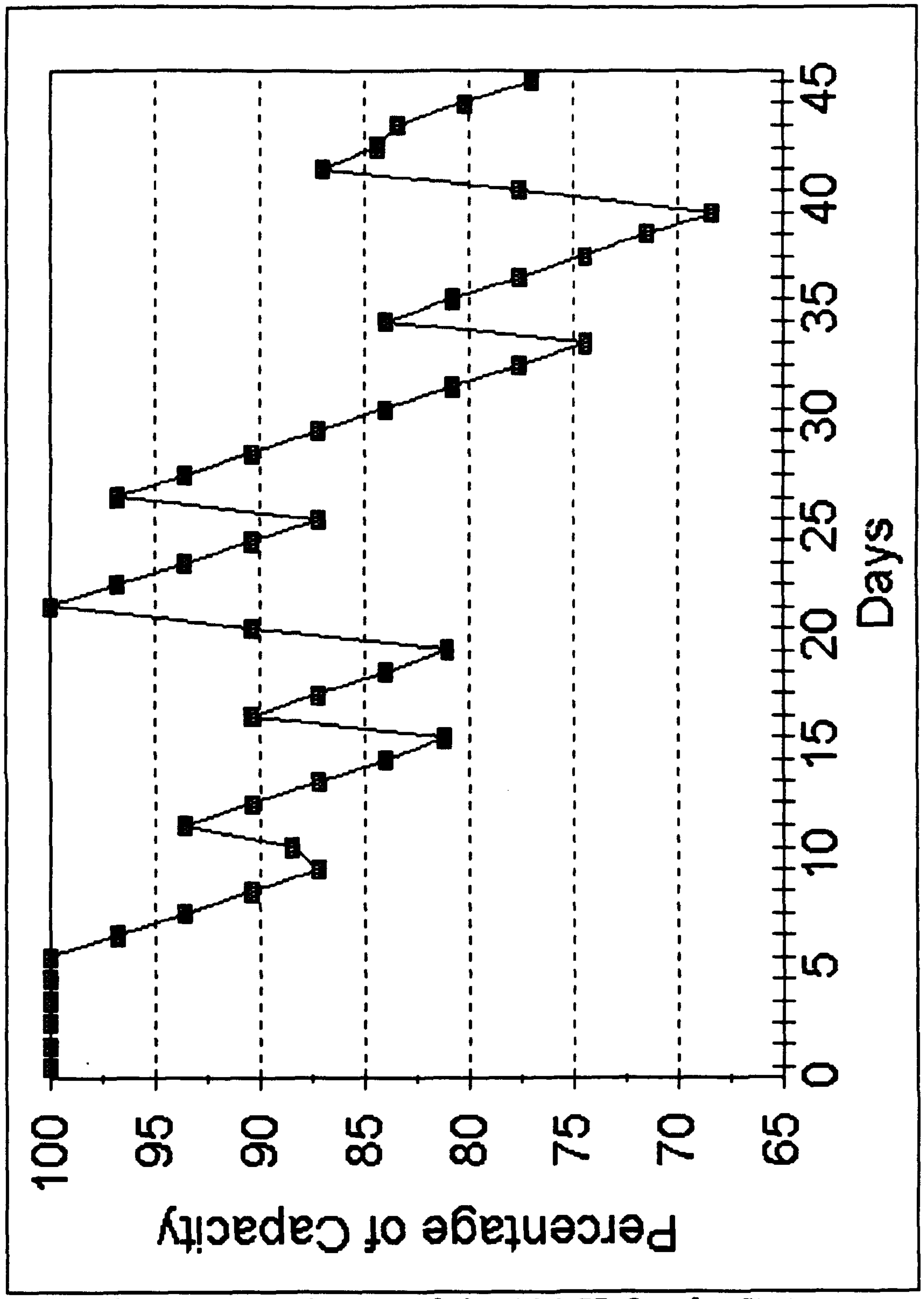


Figure 13 CTG 51.1 Alpha TOW II Supply Status

modeling a wargame logistics problem. To these ends the simulation was a success. Throughout the execution, care was taken to ensure that Commodities were indeed delivered to the Units and that deployment was handled in a reasonable fashion. At no point did a failure of the model allow Units to run out of fuel or any other critical Commodity. S3 responded to scenario events, such as the raid on CTF50, by automatically scheduling resupply of Commodities in the desired fashion. Additionally, the flexibility of the model in dealing with game events was demonstrated by the ease with which the Logistics Umpire could move Units, change their Combat Intensity, modify their Inventory position, and track their supply state. The scenario shows that S3 models logistics in an effective and reasonable manner; certainly well enough to add a sense of the logistics problems facing joint commanders today.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

It is clear that the Surge and Sustainment Simulation meets the requirements of a theater logistics simulation for wargaming set forth in Section II. S3 describes a complete wargame logistics system as needed to provide unit supply status in its inherent detail. It is sufficiently flexible to allow for game events and player decisions. Once a more sizable database is constructed, the generation of scenarios with the Scenario Editor will make using S3 a simple proposition. Finally, the current menu interface provides an acceptable means for the user to modify logistics constraints and receive the necessary results.

Of course the key advantage of the S3 is that it is based on a simple and easily understood network model of logistics. More complicated models will probably prove to be less efficient in the use of computer resources and only provide a minor improvement in detail. Course representations of logistics systems, such as abstract feasibility evaluators, give a gross estimate of unit mobilization and are useful in actual deployment planning, but do not provide the level of detail required by computer wargames.

Of course, S3 is by no means the complete solution to the wargame logistics problem. The real solution is to incorporate logistics into future naval wargame models from the outset. In the meantime, S3 will fill the gap left by wargaming systems that do not provide for logistics.

B. RECOMMENDATIONS

Despite the conclusion that the Surge and Sustainment Simulation can fill the wargaming logistics gap, the program is still in its infancy. There are many areas to recommend improvement. In its current state, S3 lacks some of the features that would significantly improve its attractiveness to naval wargamers. Some of these features include a graphics capability, a windows and dialog box user interface, reports generation, an improved database, and new object methods that more closely model logistics functions.

Graphics provide more than mere pretty pictures for the user to view while the model grinds away at its task. Graphics enhance the ability of programmers and users to assess the validity of model actions and logic. A visual representation of the simulation processes provides instant feedback and is invaluable for trouble shooting and program debugging. With the SIMGRAPHICS II package available for use with MODSIM II on MicroSoft Windows or Unix Workstations, the programmer can increase productivity in problem solving.

The SIMGRAPHICS II package includes the ability to create dialog boxes, splash screens, maps and other graphic images. The incorporation of these constructs into S3 will significantly improve the user interface. The primitive menu system that was designed specifically to allow S3 to operate in a non-window environment can be replaced with the current and more user friendly system. Since the beginning of this project, CACI has phased out its purely DOS version of MODSIM II. As a result, the user will be required to support a windows system in order to use MODSIM II. Thus, the Surge and Sustainment Simulation might as well take advantage of the available graphics package.

Currently there is no report generation available for S3. To meet the time constraints of the project, it was decided to neglect this feature. Given the nature of modern naval wargaming, it seems that this would be a necessary requirement for the final product. Both umpire and player alike will need a paper representation of the output of S3 in order to increase flexibility.

Improvements to the database structure of S3 would also be welcomed. The current flat file data structure is primitive and inefficient. One of its few advantages is that the informed user can directly edit files to change data values (although the Scenario Editor eliminates the need to do so). Its main disadvantage is that the data files for each Scenario, Base, Unit SubUnit, and Transporter are held together by a fragile relationship specified in master files that also inhabit the database. The loss or corruption of these files can mean the temporary loss of important data. Corruption of the data files will also lead to run time errors--not a desirable state of affairs. Nevertheless, the primitive database serves its purpose until a more sophisticated system can be installed.

Finally, there are several quirks in the logic of the program that will require the attention of an operational logistician. Besides the previously mentioned areas, improvement can be made in the way the Transporter Manager selects a Transporter for a given task. Additionally, the Logistics Manager could use more refinement in the logic by which it selects suppliers, picks routes, and handles requests. These improvements, though not critical to the operation of the model, will bring the output of the model more closely in line with the reality that it is attempting to describe.

APPENDIX A

LOGISTICS MODEL REVIEW REPORT OF FINDINGS

Logistic Model Review

Report of Findings

LT Robert S. Murphy
LT John A. Long

Logistic Model Review

1. The purpose of this report is to present the findings of an in-depth study of the available logistic models, with the objective of identifying a model or group of models which would provide the wargamer with a realistic logistic constraint on operations.

2. Each model was reviewed and tested based on its ability to provide the user with closure dates and a realistic logistic pipeline in an easy to learn and operate format.

3. A detailed review of the models in contained in Appendix A.

4. Based on this investigation, the Crisis Action Model (CAM) and DEPLOY are the only viable models for use. CAM is the better of the two, very easy to use and understand, but both would be acceptable to the user after a half day's training. The database for each would have to be modified slightly to fit the given scenario. The major difference lies in the operating systems. CAM runs under Microsoft Windows, which the wargaming department would have to purchase and install on each machine involved. DEPLOY runs under the normal DOS operating system. One advantage to CAM is the short turn-around time one a player's course of action is set. This would conceivably allow more than one group to use the same computer.

In the final analysis, both of these models are capable of providing the required support to the war game, and should be considered by the designer of the game for inclusion.

5. LCDR McDonaugh holds the software copies of all models reviewed.

6. Currently, the ideal model for logistics support of wargaming has not been found. The additional qualities required in such a model would include the ability to take "snapshots" of individual units, transshipment nodes, and transportation assets in order to determine their exact supply status. Visibility of critical supply items in a report form would enhance the facilitator's ability to tailor the storyline for the players. Also, game facilitators need the ability to directly affect the supply status of units and assess battle damage to the logistics pipeline. Finally, an added measure of realism would enhance the player's appreciation for logistics problems. For example, UNREP scheduling provides a realistic yet not excessively burdensome constraint on player action. The ultimate requirement is a logistics model that forces players to acknowledge the importance of logistics without significantly degenerating game play.

Respectfully Submitted,

[illegible] LT, USN

[illegible] LT, USN

Name: Crisis Action Model (CAM)

Proponent: U.S. ARMY War College

POC: Dennis Konkel DSN 242-4169

System: 386 or better; 4 Megs RAM; MS Windows

Stated Objective:

CAM is used to deploy units and supplies tు one or more theaters and determine, based upon supply levels achieved and unit closure dates whether a player's campaign is supportable. Both airlift (MAC and CRAF) and sealift (MSC,RRF, NATO ships, NDRD, etc.) are provided. Players determine the types and amount of strategic lift to be utilized.

Evaluation:

This model fulfills the basic need for closure date specification and level of supply achieved in-theater. It was created for use at the U.S. ARMY War College to validate a given operation plan, much the same role envisioned here. Of help to the planners in the game, all dates are specified as "N+", from date of notification of crisis.

Basic use starts with an introductory menu that in the beginning holds only the "Basecase" option. This option holds the database only with no movement commands (it is also from here that revisions on saved campaigns will be accessed). From here the user can pick from 29 different "policies" by number. The numbers range from 1 to 125 with unused numbers held in reserve for future expansion. These options fall into five categories: define the combat forces and strategic lift assets to be used in the scenario; mobilize the lift assets and reserve units for each service· allocate strategic lift within the theater; deploy the uni[illegible] a theater, specify supply level to maintain, and desi[illegible]e airfields and seaports; save results and print. When the player is satisfied with his plan, he executes the model and receives the closure dates and supply levels in theater. Two additional features of value to the user/player is the implementation of an easy code that displays the cause of a unit closure failure and a table that reveals a base's current throughput as a percentage of capacity available. Decisions and data can be view from a service-specific or joint aspect. Damage, in the form of a reduction in throughput or the loss of a platform, is supported. The accompanying manual states a period of one month for database creation, however given the Logistics database currently in work in the logistics branch and the rather complete existing database, this time could be reduced. This becomes an advantage, in that the database could readily be modified to fit a given scenario. Additionally, an updated version is expected that will further ease the learning curve of an already user-friendly model.

Name: Deployment Model (DEPLOY)

Proponent: National Defense University
Institute for Higher Defense Education
War Gaming and Simulation Center

POC: R.D. Wright DSN 335-1251

System: Runs on IBM and compatibles with MS-DOS

Stated Objective:

Illustrate United States strategic mobility resources; show key issues for U.S. force deployment planning; indicate limitations to U.S. response capabilities for single theater and multi-theater crisis.

Evaluation:

The model is menu driven and gives good detail as to available resources and capabilities, though highly aggregated. The data base that exists with the model seems slightly out of date, but could be revised. Basic use is through the selection of strategic lift assets and allocation, and prioritized force lists. The model can develop the data in three ways: lift required given units and RDD, Closure given units and available lift, or closure and supply levels given units and available lift. It is able to fulfill the requirements of the war gaming department. Drawbacks include: some data entry in the form of formulas, though simple, which increase the height of the learning curve; no CONUS transportation constraints, no intermediate throughput capacity, and no intra-theater lift. Additionally, the output of the model is unnecessarily complicated, but readable given some time. Overall, the model is recommended for consideration for the March student's game based upon its performance, the ability to run under DOS, the ease of use and relatively short time required to learn the system, and the ability to change the model's characteristics to meet the scenario at hand.

Name: Cady Model

Proponent: Naval War College

POC: LCDR D. McDonaugh

System: Lotus 1-2-3

Stated Objective:

Provide students playing a war game with a means to test the supportability of their plan.

Evaluation:

The Cady Model was written in Lotus 1-2-3 and is essentially a large spreadsheet that keeps a record of the requirements of the units selected. The model has no documentation, and is difficult to learn to operate. Some of the macros used were either overwritten or not completed, further limiting its effectiveness. Though the model roughly performs its mission, it's steep learning curve, rough approximations and difficulty of use (even for the initiated) makes it undesirable for use at this time.

Name: Operational Logistic Simulator (OPLOGS) Version ALPHA

Proponent: Space & Naval Warfare Systems Command

POC: Mr. Scott Lerman, APJ, Ridgefield, NJ

System: IBM and compatibles (386 or better preferred) with MS-DOS

Stated Objective:

Simplify the logistician's task by automating a resupply process and providing meaningful results in terms of the costs, time, and quantities involved.

Evaluation:

This model does a good job of representing the supply network and actions involved in moving goods. However, this is of no use to the wargamer. The expense in time for each cell to set up a distribution network is prohibitive.

Name: Rapid Deployment Exercise (RADEX)

Proponent: Air Force Wargaming Center (AFWC), Maxwell AFB, AL 36112-5532

Point of Contact: LCOL N. Coyle, AUCADRE/WGO, Maxwell AFB, AL 36112-5532

System: IBM-compatible PC with 640K RAM, 10MB Storage, MS-DOS operating system. Also Requires INGRES PC database management system for database modification.

Stated Objective:

RADEX and its sister program, JPLAN are designed to support deployment planning seminar exercises. Players input force lists and deployment phasing plans using basic JOPES concepts. The program evaluates the feasibility of those plans and reports on shortfalls.

Evaluation:

RADEX fulfills its mission as an deployment exercise driver and could be applicable to war gaming at NWC. The program provides a basic evaluation of the feasibility of a user developed deployment plan, which partially satisfies the requirements for a war gaming logistics tool. This basic evaluation consists of calculating arrival dates for units in theater and highlighting any shortfalls in the plan. Although the shortfalls are identified without explanation, they are easy to interpret given the uncomplicated format of the feasibilty reports. Building the force deployment modules is made relatively simple using the pull-down menu system. Most of the typical deployable units are provided at various levels of aggregation. The user merely has to choose which units to deploy by highlighting them on a pre-built unit list. Many features, such as a map of the AOR, are provided at the touch of a key from any location within the program. Overall, RADEX is quick to learn and user friendly.

Nevertheless, there are two major problems with the employment of RADEX as a logistics aid for NWC war games. First, the program performs only one task, the evaluation of a deployment plan. No provisions are made for sustainment, management of critical commodities, or the deployment of naval forces (although sealift is modeled). "Snapshots" of a unit's status at various times cannot be provided making it difficult for facilitators and umpires to provide any logistics information to players besides the arrival times of units in theater.

The second major difficulty is that to utilize the RADEX software for war gaming requires significant preparation of the static database. This process most likely involves the use of the INGRES database management system, which was not available for review. According to the JCS Catalog of Models, the time required for database preparation is approximately ten man-weeks.

<u>Name</u>: Petroleum, Oil, Lubricants (POL)

<u>Proponent</u>: War Gaming Department, Naval War College.

<u>Point of Contact</u>: Micromodels Manager, (401) 841-3276.

<u>System</u>: IBM-compatible PC with 512K RAM, MS-DOS operating system

<u>Stated Objective</u>:

POL models intratheater petroleum, oil, and lubricants consumption and resupply as a logistics input to a separate, parallel wargame such as ENWGS or a seminar.

<u>Evaluation</u>:

The POL model adequately models the movement of commodities from shore bases to battle groups via shuttle ships, however, this is the limit of its ability. It does not handle to movement of commodities from point to point or into the theater. The model only portrays the one level of the logistics pyramid from the forward logistic support site (FLSS) to the battle group. Some critical inaccuracies which detract from the utilization of POL as a logistic constraint on a wargame are as follows:

* High front-end investment in database creation.

* Does not model the movement of other than Class III material (although the program can be tricked into moving these commodities).

* No station ships or CONREP. All ships modeled are combatants or shuttle ships. This goes against current doctrine of having a CLF ship operating with the battle group at all times.

* No scheduling of UNREPs. Ships are automatically brought to full supply strength without the need to pull off line to UNREP.

* Shuttle ships only move from base to battle group, not between groups.

* No modeling of VERTREP or COD methods of resupply.

Finally, the software was erratic and crash-prone. Simple error detection (in this case, division by zero) was not provided for the version that was evaluated. Given the high front end investment required for database entry, this fact alone suggests the abandonment of this version of the model.

Name: Air Availability and Repair (AAR)

Proponent: War Gaming Department, Naval War College

Point of Contact: Micromodels Manager, (401) 841-3276, DSN 948-3276.

System: IBM compatible PC with 512K RAM, MS-DOS operating system.

Stated Objective:

AAR models aircraft battle damage assessment and repair based on user inputs to a program managed database. Movement of assets and repair kits is modeled in a limited fashion. The model has been specifically designed as an input to a separate wargame (ENWGS or seminar) run in parallel with AAR.

Evaluation:

Essentially, the AAR model is a battle damage calculator with very few functions that require the power of a computer. The program determines the number of aircraft lost, damaged, and repairable based solely on user inputs to the database and a Monte Carlo evaluation of probabilities. The creation of the database takes much of the user's time and is almost as complicated as entering missions on the ENWGS system.

Given the apparent method underlying the model's execution of aircraft battle damage assessment and repair, there is nothing handled by the program that cannot not be determined by an human umpire during an ENWGS game. The umpires can determine by mission, how many aircraft are lost, how many are repairable, and how long repairs will take. Additionally, the human BDA and repair system is simpler and allows a great deal of flexibility in the determination of ground truth, something the program does not allow.

In a larger sense, the AAR model does not address any other issues critical to the treatment of logistics for a wargame. While it is possible to model a larger picture by running other programs that handle different logistics needs in parallel with AAR, the results of AAR are not worth the extensive time involved for database creation and manipulation. A more general model could accomplish much more.

Name: Medical Regulating Model (MRM)

Proponent: War Gaming Department, Naval War College.

Point of Contact: Micromodels Manager, (401) 841-3276, DSN 948-3276.

System: IBM compatible PC with 512K RAM, MS-DOS operating system.

Stated Objective:

MRM models combat zone casualties and return to duty rates in support of a separate, parallel ENWGS or seminar game. Evacuation of casualties to communication zone facilities and CONUS is also modeled.

Evaluation:

MRM goes slightly beyond the calulator-like determinism of the Air Availablity and Repair (AAR) model by automatically moving casualties to rear areas (communication zone or CONUS). The advantage of the model lies in the ability to execute user directed evacuation policy and evaluate the feasibility of a medical plan given an expected casualty rates. However, if the results of medical treatment and return to duty rates are not important to the main wargame, it is recommended that MRM not be utilized since the front end costs in database creation are so high. The rates for KIA, WIA, DOW (died of wounds), and DNBI (disease, non-battle injury) are determined by the user, however, the program defaults to a standard set of rates that are predetermined. Still there remains considerable user input to the dynamic portion of the database in order for the program to model a specific scenario.

Another drawback to MRM is that the program is designed to handle land combat casualties, which tend to come at a fairly steady rate depending on warfare intensity. In naval warfare, casualties tend to come as a result of catastrophic events, such as the sinking of a ship. The model does not provide an easy method to inflict mass casualties directly on a "population at risk" on a one-time basis.

APPENDIX B

SWOS-129 COMMODITY FILES

```
NumberOfCommodities: 62

Name: Mk-82 Class: Ammo ProduceAt: 250.00
Length: 72.00 Width: 24.00 Height: 24.00
Weight: 500.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: Water Class: Fuel ProduceAt: 1000000.00
Length: 0.00 Width: 0.00 Height: 0.00
Weight: 300.00 Priority: 5 EmerPriority: 4
OutSize: FALSE

Name: Personnel Class: Personnel ProduceAt: 1000.00
Length: 18.00 Width: 18.00 Height: 72.00
Weight: 400.00 Priority: 1 EmerPriority: 1
OutSize: FALSE

Name: F44 Class: Fuel ProduceAt: 1000000000.00
Length: 0.00 Width: 0.00 Height: 0.00
Weight: 264.00 Priority: 5 EmerPriority: 4
OutSize: FALSE

Name: AIM-9M Class: Ammo ProduceAt: 80.00
Length: 60.00 Width: 12.00 Height: 12.00
Weight: 200.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: AIM-7M Class: Ammo ProduceAt: 10.00
Length: 84.00 Width: 20.00 Height: 20.00
Weight: 400.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: General Class: Other ProduceAt: 100000000.00
Length: 12.00 Width: 12.00 Height: 12.00
Weight: 134.00 Priority: 5 EmerPriority: 1
OutSize: FALSE

Name: F-15C Class: Major ProduceAt: 0.00
Length: 900.00 Width: 900.00 Height: 900.00
Weight: 20300.00 Priority: 1 EmerPriority: 1
OutSize: TRUE

Name: M-1A1 Class: Major ProduceAt: 0.50
Length: 400.00 Width: 180.00 Height: 96.00
Weight: 120000.00 Priority: 4 EmerPriority: 1
OutSize: TRUE

Name: F/A-18C Class: Major ProduceAt: 0.25
Length: 600.00 Width: 600.00 Height: 132.00
Weight: 20000.00 Priority: 6 EmerPriority: 1
OutSize: TRUE

Name: F76 Class: Fuel ProduceAt: 2000000.00
Length: 0.00 Width: 0.00 Height: 0.00
Weight: 308.00 Priority: 5 EmerPriority: 4
OutSize: FALSE

Name: AIM-54C Class: Ammo ProduceAt: 1.00
Length: 60.00 Width: 24.00 Height: 24.00
```

```
Weight: 500.00 Priority: 1 EmerPriority: 1
OutSize: FALSE

Name: FFV Class: FFV ProduceAt: 1000.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 250.00 Priority: 5 EmerPriority: 1
OutSize: FALSE

Name: DRY Class: FFV ProduceAt: 1000.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 250.00 Priority: 5 EmerPriority: 1
OutSize: FALSE

Name: FROZEN Class: FFV ProduceAt: 1000.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 500.00 Priority: 5 EmerPriority: 1
OutSize: FALSE

Name: Mines Class: Ammo ProduceAt: 200.00
Length: 50.00 Width: 24.00 Height: 24.00
Weight: 500.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: Mail Class: Other ProduceAt: 500.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 500.00 Priority: 6 EmerPriority: 1
OutSize: FALSE

Name: 20MM Class: Ammo ProduceAt: 100.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 20MM/76 Class: Ammo ProduceAt: 100.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: RIM-7 Class: Ammo ProduceAt: 20.00
Length: 84.00 Width: 24.00 Height: 24.00
Weight: 800.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: SRBOC Class: Ammo ProduceAt: 200.00
Length: 36.00 Width: 8.00 Height: 8.00
Weight: 50.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: TLAM-N Class: Ammo ProduceAt: 20.00
Length: 144.00 Width: 36.00 Height: 36.00
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: TLAM-D Class: Ammo ProduceAt: 20.00
Length: 144.00 Width: 36.00 Height: 36.00
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: TLAM-C Class: Ammo ProduceAt: 20.00
Length: 144.00 Width: 36.00 Height: 36.00
```

```
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: TASM Class: Ammo ProduceAt: 20.00
Length: 144.00 Width: 36.00 Height: 36.00
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: HARPOON-S Class: Ammo ProduceAt: 40.00
Length: 120.00 Width: 24.00 Height: 24.00
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: HARPOON-A Class: Ammo ProduceAt: 40.00
Length: 120.00 Width: 24.00 Height: 24.00
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: ASROC Class: Ammo ProduceAt: 10.00
Length: 12.00 Width: 24.00 Height: 24.00
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: Mk-46 Class: Ammo ProduceAt: 10.00
Length: 96.00 Width: 16.00 Height: 16.00
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: SM-2ER Class: Ammo ProduceAt: 20.00
Length: 120.00 Width: 16.00 Height: 16.00
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: SM-2MR Class: Ammo ProduceAt: 20.00
Length: 120.00 Width: 16.00 Height: 16.00
Weight: 1800.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 127MM/54 Class: Ammo ProduceAt: 50.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 76MM/62 Class: Ammo ProduceAt: 50.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 25MM Class: Ammo ProduceAt: 100.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 76MM/50 Class: Ammo ProduceAt: 20.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: HARM Class: Ammo ProduceAt: 20.00
Length: 96.00 Width: 12.00 Height: 12.00
```

```
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: AMRAAM Class: Ammo ProduceAt: 30.00
Length: 84.00 Width: 16.00 Height: 16.00
Weight: 1500.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: AGM-62 Class: Ammo ProduceAt: 10.00
Length: 72.00 Width: 24.00 Height: 24.00
Weight: 1500.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: AGM-65 Class: Ammo ProduceAt: 30.00
Length: 72.00 Width: 24.00 Height: 24.00
Weight: 1500.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: SONOBUOY Class: Ammo ProduceAt: 2000.00
Length: 60.00 Width: 8.00 Height: 8.00
Weight: 50.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: DEPTHCHARGE Class: Ammo ProduceAt: 10.00
Length: 60.00 Width: 36.00 Height: 36.00
Weight: 2000.00 Priority: 5 EmerPriority: 1
OutSize: FALSE

Name: PENGUIN Class: Ammo ProduceAt: 25.00
Length: 84.00 Width: 24.00 Height: 24.00
Weight: 1500.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: PAVEWAY Class: Ammo ProduceAt: 50.00
Length: 84.00 Width: 16.00 Height: 16.00
Weight: 750.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: AIM-9L Class: Ammo ProduceAt: 30.00
Length: 60.00 Width: 12.00 Height: 12.00
Weight: 200.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: TOW Class: Ammo ProduceAt: 100.00
Length: 60.00 Width: 12.00 Height: 12.00
Weight: 100.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: HELLFIRE Class: Ammo ProduceAt: 50.00
Length: 72.00 Width: 12.00 Height: 12.00
Weight: 194.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 40MMGREN Class: Ammo ProduceAt: 200.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: Mk-83 Class: Ammo ProduceAt: 200.00
Length: 72.00 Width: 24.00 Height: 24.00
```

```
Weight: 750.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: Mk-84 Class: Ammo ProduceAt: 200.00
Length: 72.00 Width: 24.00 Height: 24.00
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: ROCKEYE Class: Ammo ProduceAt: 200.00
Length: 72.00 Width: 24.00 Height: 24.00
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: FAE Class: Ammo ProduceAt: 200.00
Length: 72.00 Width: 24.00 Height: 24.00
Weight: 1000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 155MM Class: Ammo ProduceAt: 100.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 105MM Class: Ammo ProduceAt: 150.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 81MM Class: Ammo ProduceAt: 300.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 60MM Class: Ammo ProduceAt: 300.00
Length: 48.00 Width: 40.00 Height: 72.00
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: SM-1MR Class: Ammo ProduceAt: 20.00
Length: 108.00 Width: 16.00 Height: 16.00
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: DIESEL Class: Fuel ProduceAt: 2000000.00
Length: 0.00 Width: 0.00 Height: 0.00
Weight: 0.00 Priority: 4 EmerPriority: 4
OutSize: FALSE

Name: AH-64 Class: Major ProduceAt: 0.50
Length: 591.00 Width: 195.00 Height: 150.00
Weight: 10505.00 Priority: 4 EmerPriority: 1
OutSize: TRUE

Name: UNITEQ Class: Major ProduceAt: 10000.00
Length: 48.00 Width: 40.00 Height: 96.00
Weight: 2000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: 105HOW Class: Major ProduceAt: 10.00
Length: 120.00 Width: 72.00 Height: 60.00
```

```
Weight: 4000.00 Priority: 4 EmerPriority: 1
OutSize: FALSE

Name: M-109 Class: Major ProduceAt: 0.50
Length: 400.00 Width: 180.00 Height: 96.00
Weight: 80000.00 Priority: 4 EmerPriority: 1
OutSize: TRUE

Name: MOGAS Class: Fuel ProduceAt: 2000000.00
Length: 0.00 Width: 0.00 Height: 0.00
Weight: 0.00 Priority: 4 EmerPriority: 4
OutSize: FALSE
```

APPENDIX C

SWOS-129 TRANSPORTER FILES

```
NumberOfFiles: 17

==========================================================================
C-5A.dat
==========================================================================

Type: C-5A
Class: Aircraft
SubClass: General
Length: 248.00 Width: 222.00
MaxSpeed: 450.00 MaxRange: 1500.00

MaxCargoArea: 219581.00 MaxCargoCube: 18368.00 MaxCargoWeight: 242400.00
MaxCargoLength: 1452.00 MaxCargoWidth: 220.00 MaxCargoHeight: 156.00
 MaxPax: 72.00 MaxGas: 0.00
OverSize: TRUE

==========================================================================
==========================================================================

Type: AOE-6
Class: Ship
SubClass: BreakBulk
Length: 800.00 Width: 150.00
MaxSpeed: 20.00 MaxRange: 4500.00

MaxCargoArea: 219581.00 MaxCargoCube: 1756650.00 MaxCargoWeight: 132408974.00
MaxCargoLength: 1000.00 MaxCargoWidth: 500.00 MaxCargoHeight: 120.00
 MaxPax: 100.00 MaxGas: 100000.00
OverSize: TRUE

==========================================================================
==========================================================================

Type: TRUCKCONVOY
Class: Truck
SubClass: BreakBulk
Length: 10.00 Width: 8.00
MaxSpeed: 40.00 MaxRange: 3000.00

MaxCargoArea: 3200.00 MaxCargoCube: 13420.00 MaxCargoWeight: 240000.00
MaxCargoLength: 480.00 MaxCargoWidth: 200.00 MaxCargoHeight: 96.00
 MaxPax: 200.00 MaxGas: 500.00
OverSize: TRUE

==========================================================================
==========================================================================

Type: C-141B
Class: Aircraft
SubClass: General
Length: 168.00 Width: 160.00
MaxSpeed: 425.00 MaxRange: 1500.00

MaxCargoArea: 878.00 MaxCargoCube: 7024.00 MaxCargoWeight: 90200.00
MaxCargoLength: 1120.00 MaxCargoWidth: 123.00 MaxCargoHeight: 109.00
 MaxPax: 0.00 MaxGas: 0.00
OverSize: FALSE

==========================================================================
================================================================================

Type: B747
Class: Aircraft
SubClass: Pax
Length: 300.00 Width: 350.00
MaxSpeed: 350.00 MaxRange: 3500.00

MaxCargoArea: 5000.00 MaxCargoCube: 40000.00 MaxCargoWeight: 400000.00
MaxCargoLength: 188.00 MaxCargoWidth: 108.00 MaxCargoHeight: 100.00
 MaxPax: 395.00 MaxGas: 0.00
OverSize: FALSE

================================================================================
================================================================================

Type: PAXTRAIN
Class: Rail
SubClass: Pax
Length: 21.00 Width: 10.00
MaxSpeed: 50.00 MaxRange: 3000.00

MaxCargoArea: 500.00 MaxCargoCube: 4000.00 MaxCargoWeight: 500000.00
MaxCargoLength: 120.00 MaxCargoWidth: 96.00 MaxCargoHeight: 96.00
 MaxPax: 770.00 MaxGas: 0.00
OverSize: FALSE

================================================================================
================================================================================

Type: AO177
Class: Ship
SubClass: Liquid
Length: 800.00 Width: 100.00
MaxSpeed: 19.00 MaxRange: 3500.00

MaxCargoArea: 0.00 MaxCargoCube: 0.00 MaxCargoWeight: 0.00
MaxCargoLength: 0.00 MaxCargoWidth: 0.00 MaxCargoHeight: 0.00
 MaxPax: 5.00 MaxGas: 200000.00
OverSize: FALSE

================================================================================
================================================================================

Type: AE27
Class: Ship
SubClass: BreakBulk
Length: 560.00 Width: 80.00
MaxSpeed: 20.00 MaxRange: 6000.00

MaxCargoArea: 49116.00 MaxCargoCube: 491166.00 MaxCargoWeight: 24558300.00
MaxCargoLength: 500.00 MaxCargoWidth: 500.00 MaxCargoHeight: 120.00
 MaxPax: 0.00 MaxGas: 1000.00
OverSize: TRUE

================================================================================
================================================================================

Type: Breakbulk
Class: Ship
```

```
SubClass: BreakBulk
Length: 490.00 Width: 80.00
MaxSpeed: 17.70 MaxRange: 10000.00

MaxCargoArea: 59149.00 MaxCargoCube: 602120.00 MaxCargoWeight: 36127200.00
MaxCargoLength: 500.00 MaxCargoWidth: 500.00 MaxCargoHeight: 120.00
 MaxPax: 10.00 MaxGas: 1000.00
OverSize: TRUE

================================================================================
================================================================================

Type: C4-S-1H
Class: Ship
SubClass: BreakBulk
Length: 565.00 Width: 76.00
MaxSpeed: 20.00 MaxRange: 10000.00

MaxCargoArea: 65605.00 MaxCargoCube: 508710.00 MaxCargoWeight: 34000000.00
MaxCargoLength: 500.00 MaxCargoWidth: 500.00 MaxCargoHeight: 120.00
 MaxPax: 20.00 MaxGas: 1000.00
OverSize: TRUE

================================================================================
================================================================================

Type: TANKER
Class: Ship
SubClass: Liquid
Length: 800.00 Width: 100.00
MaxSpeed: 20.00 MaxRange: 10000.00

MaxCargoArea: 0.00 MaxCargoCube: 0.00 MaxCargoWeight: 0.00
MaxCargoLength: 0.00 MaxCargoWidth: 0.00 MaxCargoHeight: 0.00
 MaxPax: 0.00 MaxGas: 200000.00
OverSize: FALSE

================================================================================
================================================================================

Type: UNITTRAIN
Class: Rail
SubClass: RoRo
Length: 50.00 Width: 11.00
MaxSpeed: 22.00 MaxRange: 5000.00

MaxCargoArea: 28350.00 MaxCargoCube: 274350.00 MaxCargoWeight: 10000000.00
MaxCargoLength: 648.00 MaxCargoWidth: 126.00 MaxCargoHeight: 117.00
 MaxPax: 0.00 MaxGas: 0.00
OverSize: TRUE

================================================================================
================================================================================

Type: FREIGHTTRAIN
Class: Rail
SubClass: BreakBulk
Length: 30.00 Width: 10.50
MaxSpeed: 13.00 MaxRange: 5000.00
```

```
MaxCargoArea: 16879.00 MaxCargoCube: 164610.00 MaxCargoWeight: 4920000.00
MaxCargoLength: 717.00 MaxCargoWidth: 113.00 MaxCargoHeight: 117.00
 MaxPax: 10.00 MaxGas: 476.19
OverSize: TRUE

==============================================================================
==============================================================================

Type: TANKERTRAIN
Class: Rail
SubClass: Liquid
Length: 30.00 Width: 126.00
MaxSpeed: 13.00 MaxRange: 5000.00

MaxCargoArea: 0.00 MaxCargoCube: 0.00 MaxCargoWeight: 0.00
MaxCargoLength: 0.00 MaxCargoWidth: 0.00 MaxCargoHeight: 0.00
 MaxPax: 0.00 MaxGas: 14285.00
OverSize: FALSE

==============================================================================
==============================================================================

Type: TANKTRUCKS
Class: Truck
SubClass: Liquid
Length: 10.00 Width: 8.00
MaxSpeed: 20.00 MaxRange: 5000.00

MaxCargoArea: 0.00 MaxCargoCube: 0.00 MaxCargoWeight: 0.00
MaxCargoLength: 0.00 MaxCargoWidth: 0.00 MaxCargoHeight: 0.00
 MaxPax: 0.00 MaxGas: 1190.00
OverSize: FALSE

==============================================================================
==============================================================================

Type: Comet
Class: Ship
SubClass: RoRo
Length: 647.00 Width: 78.00
MaxSpeed: 18.00 MaxRange: 6000.00

MaxCargoArea: 86478.00 MaxCargoCube: 683840.00 MaxCargoWeight: 41030400.00
MaxCargoLength: 600.00 MaxCargoWidth: 600.00 MaxCargoHeight: 120.00
 MaxPax: 10.00 MaxGas: 100.00
OverSize: TRUE

==============================================================================
==============================================================================

Type: SL-7
Class: Ship
SubClass: RoRo
Length: 946.00 Width: 106.00
MaxSpeed: 27.00 MaxRange: 6000.00

MaxCargoArea: 217600.00 MaxCargoCube: 2162440.00 MaxCargoWeight: 129746400.00
MaxCargoLength: 600.00 MaxCargoWidth: 600.00 MaxCargoHeight: 120.00
MaxPax: 10.00 MaxGas: 100.00
OverSize: TRUE
```

APPENDIX D

SWOS-129 SUBUNIT FILES

```
FA-18E
Class:  Air

Commodities: 2

AIM-9
StockTo 0.00 Deployment: FALSE
HighRate: 4.000 MedRate 2.000 LowRate 1.000 NoneRate 0.000

JP-5
StockTo 0.00 Deployment: TRUE
HighRate: 1000.000 MedRate 800.000 LowRate 400.000 NoneRate 400.000
```

```
F-14D
Class:  Air

Commodities: 7

20MM
StockTo 0.00 Deployment: FALSE
HighRate: 0.008 MedRate 0.008 LowRate 0.008 NoneRate 0.000

AIM-54C
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARM
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARPOON-A
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AIM-7M
StockTo 0.00 Deployment: FALSE
HighRate: 0.280 MedRate 0.280 LowRate 0.280 NoneRate 0.000

AIM-9M
StockTo 0.00 Deployment: FALSE
HighRate: 0.230 MedRate 0.230 LowRate 0.100 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 170.500 MedRate 129.500 LowRate 129.500 NoneRate 129.500
```

```
A-6E
Class:  Air

Commodities: 7

Mk-82
StockTo 0.00 Deployment: FALSE
HighRate: 1.400 MedRate 1.400 LowRate 1.400 NoneRate 0.000

Mk-84
StockTo 0.00 Deployment: FALSE
HighRate: 0.600 MedRate 0.600 LowRate 0.600 NoneRate 0.000

AGM-65
StockTo 0.00 Deployment: FALSE
HighRate: 1.100 MedRate 1.100 LowRate 1.100 NoneRate 0.000

HARPOON-A
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mines
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARM
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 127.300 MedRate 91.000 LowRate 91.000 NoneRate 68.000
```

```
EA-6B
Class:  Air

Commodities: 2

HARM
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 127.300 MedRate 91.000 LowRate 91.000 NoneRate 68.000
```

```
E-2C
Class:  Air

Commodities: 1

F44
StockTo 0.00 Deployment: FALSE
HighRate: 156.250 MedRate 100.000 LowRate 100.000 NoneRate 100.000
```

```
S-3B
Class:  Air

Commodities: 5

SONOBUOY
StockTo 0.00 Deployment: FALSE
HighRate: 60.000 MedRate 60.000 LowRate 30.000 NoneRate 0.000

DEPTHCHARGE
StockTo 0.00 Deployment: FALSE
HighRate: 2.000 MedRate 1.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 0.00 Deployment: FALSE
HighRate: 4.000 MedRate 2.000 LowRate 1.000 NoneRate 0.000

HARPOON-A
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 109.900 MedRate 90.910 LowRate 90.910 NoneRate 72.730
```

```
ES-3
Class:  Air

Commodities: 1

F44
StockTo 0.00 Deployment: FALSE
HighRate: 109.900 MedRate 90.910 LowRate 90.910 NoneRate 72.730
```

```
SH-60B
Class:  Air

Commodities: 4

Mk-46
StockTo 0.00 Deployment: FALSE
HighRate: 0.500 MedRate 0.250 LowRate 0.000 NoneRate 0.000

SONOBUOY
StockTo 0.00 Deployment: FALSE
HighRate: 25.000 MedRate 25.000 LowRate 10.000 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 40.230 MedRate 26.000 LowRate 26.000 NoneRate 26.000

PENGUIN
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
SH-60F
Class:  Air

Commodities: 3

Mk-46
StockTo 0.00 Deployment: FALSE
HighRate: 0.500 MedRate 0.250 LowRate 0.000 NoneRate 0.000

SONOBUOY
StockTo 0.00 Deployment: FALSE
HighRate: 50.000 MedRate 50.000 LowRate 25.000 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 40.230 MedRate 26.000 LowRate 26.000 NoneRate 26.000
```

```
P-3C
Class:  Air

Commodities: 5

HARPOON-A
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 0.00 Deployment: FALSE
HighRate: 2.000 MedRate 1.000 LowRate 0.000 NoneRate 0.000

SONOBUOY
StockTo 0.00 Deployment: FALSE
HighRate: 100.000 MedRate 100.000 LowRate 50.000 NoneRate 0.000

Mines
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 276.140 MedRate 276.140 LowRate 207.100 NoneRate 207.100
```

```
EP-3
Class:  Air

Commodities: 1

F44
StockTo 0.00 Deployment: FALSE
HighRate: 207.100 MedRate 207.100 LowRate 207.100 NoneRate 207.100
```

```
E-3B
Class:  Air

Commodities: 1

F44
StockTo 0.00 Deployment: FALSE
HignRate: 818.180 MedRate 818.180 LowRate 613.640 NoneRate 613.640
```

```
F-15C
Class:  Air

Commodities: 4

20MM
StockTo 0.00 Deployment: FALSE
HighRate: 0.008 MedRate 0.008 LowRate 0.008 NoneRate 0.000

AIM-7M
StockTo 0.00 Deployment: FALSE
HighRate: 0.280 MedRate 0.280 LowRate 0.280 NoneRate 0.000

AIM-9M
StockTo 0.00 Deployment: FALSE
HighRate: 0.230 MedRate 0.230 LowRate 0.230 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 204.550 MedRate 115.910 LowRate 115.910 NoneRate 115.910
```

```
F-15E
Class:  Air

Commodities: 6

20MM
StockTo 0.00 Deployment: FALSE
HighRate: 0.008 MedRate 0.008 LowRate 0.008 NoneRate 0.000

Mk-82
StockTo 0.00 Deployment: FALSE
HighRate: 2.200 MedRate 2.200 LowRate 2.200 NoneRate 0.000

AGM-65
StockTo 0.00 Deployment: FALSE
HighRate: 0.100 MedRate 0.100 LowRate 0.100 NoneRate 0.000

ROCKEYE
StockTo 0.00 Deployment: FALSE
HighRate: 0.600 MedRate 0.600 LowRate 0.600 NoneRate 0.000

PAVEWAY
StockTo 0.00 Deployment: FALSE
HighRate: 0.500 MedRate 0.500 LowRate 0.500 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 204.550 MedRate 115.910 LowRate 115.910 NoneRate 115.910
```

```
AV-8B
Class:  Air

Commodities: 6

AIM-9L
StockTo 0.00 Deployment: FALSE
HighRate: 0.230 MedRate 0.230 LowRate 0.230 NoneRate 0.000

AGM-65
StockTo 0.00 Deployment: FALSE
HighRate: 0.100 MedRate 0.100 LowRate 0.100 NoneRate 0.000

AGM-62
StockTo 0.00 Deployment: FALSE
HighRate: 0.100 MedRate 0.100 LowRate 0.100 NoneRate 0.000

PAVEWAY
StockTo 0.00 Deployment: FALSE
HighRate: 0.500 MedRate 0.500 LowRate 0.500 NoneRate 0.000

25MM
StockTo 0.00 Deployment: FALSE
HighRate: 0.007 MedRate 0.007 LowRate 0.007 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 81.820 MedRate 81.820 LowRate 40.910 NoneRate 40.910
```

```
UH-1N
Class:  Air

Commodities: 1

F44
StockTo 0.00 Deployment: FALSE
HighRate: 24.550 MedRate 16.360 LowRate 16.360 NoneRate 16.360
```

```
CH-46E
Class:  Air

Commodities: 1

F44
StockTo 0.00 Deployment: FALSE
HighRate: 38.180 MedRate 38.180 LowRate 25.450 NoneRate 25.450
```

CH-53E
Class: Air

Commodities: 1

F44
StockTo 0.00 Deployment: FALSE
HighRate: 116.880 MedRate 116.880 LowRate 77.920 NoneRate 77.920

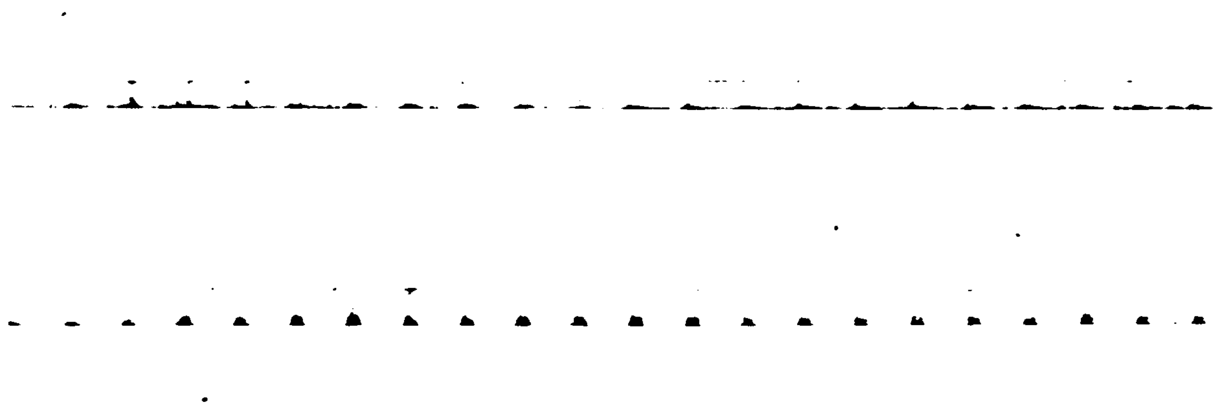
```
AH-1W
Class:  Air

Commodities: 6

TOW
StockTo 0.00 Deployment: FALSE
HighRate: 8.000 MedRate 8.000 LowRate 2.000 NoneRate 0.000

HELLFIRE
StockTo 0.00 Deployment: FALSE
HighRate: 4.000 MedRate 4.000 LowRate 2.000 NoneRate 0.000

AIM-9L
StockTo 0.00 Deployment: FALSE
HighRate: 0.230 MedRate 0.100 LowRate 0.100 NoneRate 0.000

20MM
StockTo 0.00 Deployment: FALSE
HighRate: 0.010 MedRate 0.010 LowRate 0.010 NoneRate 0.000

40MMGREN
StockTo 0.00 Deployment: FALSE
HighRate: 0.002 MedRate 0.002 LowRate 0.002 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 24.550 MedRate 24.550 LowRate 16.360 NoneRate 16.360
```

```
AH-64
Class:  Air

Commodities: 5

30MM
StockTo 0.00 Deployment: FALSE
HighRate: 0.280 MedRate 0.200 LowRate 0.120 NoneRate 0.000

HELLFIRE
StockTo 0.00 Deployment: FALSE
HighRate: 16.800 MedRate 11.800 LowRate 7.000 NoneRate 0.000

2.75INCH
StockTo 0.00 Deployment: FALSE
HighRate: 0.190 MedRate 0.140 LowRate 0.084 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 26.430 MedRate 17.620 LowRate 17.620 NoneRate 8.810

AH-64
StockTo 1.00 Deployment: TRUE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
CG47
Class:  Sea

Commodities: 9

HARPOON-S
StockTo 8.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SM-2MR
StockTo 68.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

ASROC
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

127MM/54
StockTo 50.00 Deployment: FALSE
HighRate: 10.000 MedRate 5.000 LowRate 2.000 NoneRate 0.000

20MM/76
StockTo 6.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 36.00 Deployment: FALSE
HighRate: 5.000 MedRate 3.000 LowRate 1.000 NoneRate 0.000

SRBOC
StockTo 25.00 Deployment: FALSE
HighRate: 10.000 MedRate 3.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 14285.00 Deployment: FALSE
HighRate: 714.000 MedRate 714.000 LowRate 714.000 NoneRate 714.000

F44
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

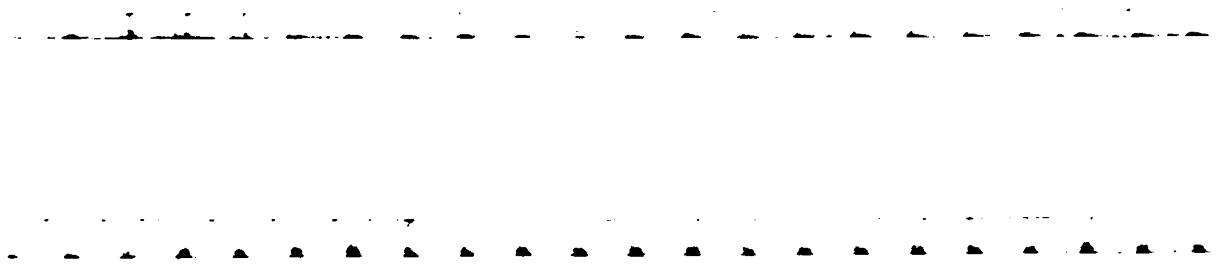
```
CG52
Class:  Sea

Commodities: 14

TLAM-N
StockTo 2.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TLAM-D
StockTo 10.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TLAM-C
StockTo 10.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TASM
StockTo 10.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARPOON-S
StockTo 8.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SM-2MR
StockTo 122.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

ASROC
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

127MM/54
StockTo 50.00 Deployment: FALSE
HighRate: 10.000 MedRate 5.000 LowRate 1.000 NoneRate 0.000

20MM/76
StockTo 6.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 36.00 Deployment: FALSE
HighRate: 5.000 MedRate 3.000 LowRate 1.000 NoneRate 0.000

SRBOC
StockTo 40.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 14285.00 Deployment: FALSE
HighRate: 714.000 MedRate 714.000 LowRate 714.000 NoneRate 714.000

F44
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SONOBUOY
StockTo 230.00 Deployment: FALSE
```

```
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```



```
DD963
Class:  Sea

Commodities: 14

TLAM-N
StockTo 2.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TLAM-D
StockTo 5.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TLAM-C
StockTo 5.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TASM
StockTo 5.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARPOON-S
StockTo 8.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

RIM-7
StockTo 24.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

ASROC
StockTo 24.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

127MM/54
StockTo 12.00 Deployment: FALSE
HighRate: 10.000 MedRate 5.000 LowRate 1.000 NoneRate 0.000

20MM/76
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 14.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SRBOC
StockTo 40.00 Deployment: FALSE
HighRate: 5.000 MedRate 2.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 11905.00 Deployment: FALSE
HighRate: 883.000 MedRate 883.000 LowRate 883.000 NoneRate 883.000

F44
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SONOBUOY
StockTo 230.00 Deployment: FALSE
```

HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000



```
FFG7
Class:  Sea

Commodities: 7

HARPOON-S
StockTo 4.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SM-1MR
StockTo 36.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

76MM/62
StockTo 50.00 Deployment: FALSE
HighRate: 10.000 MedRate 5.000 LowRate 2.000 NoneRate 0.000

20MM/76
StockTo 3.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 24.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 4285.00 Deployment: FALSE
HighRate: 428.000 MedRate 428.000 LowRate 428.000 NoneRate 428.000

F44
StockTo 452.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
CVN68
Class:  Sea

Commodities: 16

F44
StockTo 60119.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

20MM
StockTo 12.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

RIM-7
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SRBOC
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARPOON-A
StockTo 30.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 30.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SONOBUOY
StockTo 2550.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-82
StockTo 1000.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-83
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-84
StockTo 50.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARM
StockTo 40.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AGM-62
StockTo 40.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

ROCKEYE
StockTo 300.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AIM-7M
StockTo 100.00 Deployment: FALSE
```

```
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AIM-9M
StockTo 150.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AIM-54C
StockTo 50.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
LHD1
Class:  Sea

Commodities: 15

RIM-7
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

20MM/76
StockTo 6.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SRBOC
StockTo 80.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-82
StockTo 100.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AIM-9L
StockTo 50.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AGM-65
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AGM-62
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

PAVEWAY
StockTo 100.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 30000.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 14285.00 Deployment: FALSE
HighRate: 1000.000 MedRate 1000.000 LowRate 1000.000 NoneRate 1000.000

25MM
StockTo 4.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TOW
StockTo 1000.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HELLFIRE
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

20MM
StockTo 2.00 Deployment: FALSE
```

```
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

40MMGREN
StockTo 1.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
DDG51
Class:  Sea

Commodities: 10

SM-2MR
StockTo 60.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HARPOON-S
StockTo 8.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

Mk-46
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TLAM-C
StockTo 10.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TLAM-D
StockTo 0.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

127MM/54
StockTo 12.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

20MM/76
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SRBOC
StockTo 40.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 11905.00 Deployment: FALSE
HighRate: 883.000 MedRate 883.000 LowRate 883.000 NoneRate 883.000

F44
StockTo 452.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
LHA1
Class:  Sea

Commodities: 15

127MM/54
StockTo 12.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

20MM/76
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SRBOC
StockTo 40.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

25MM
StockTo 4.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AIM-9L
StockTo 50.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AGM-65
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

AGM-62
StockTo 20.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

PAVEWAY
StockTo 100.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 30000.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 14285.00 Deployment: FALSE
HighRate: 1000.000 MedRate 1000.000 LowRate 1000.000 NoneRate 1000.000

Mk-82
StockTo 100.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

TOW
StockTo 1000.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

HELLFIRE
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

20MM
StockTo 2.00 Deployment: FALSE
```

```
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

40MMGREN
StockTo 1.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
LSD41
Class:  Sea

Commodities: 4

20MM/76
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

SRBOC
StockTo 40.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 14285.00 Deployment: FALSE
HighRate: 1000.000 MedRate 1000.000 LowRate 1000.000 NoneRate 1000.000
```

```
AGF3
Class:  Sea

Commodities: 3

20MM/76
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F44
StockTo 500.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000

F76
StockTo 14285.00 Deployment: FALSE
HighRate: 1000.000 MedRate 1000.000 LowRate 1000.000 NoneRate 1000.000
```

```
MCM1
Class:  Sea

Commodities: 1

F76
StockTo 795.00 Deployment: FALSE
HighRate: 70.000 MedRate 70.000 LowRate 70.000 NoneRate 70.000
```

```
M1PLT
Class:  Land

Commodities: 2

M-1A1
StockTo 4.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 1.000

Personnel
StockTo 16.00 Deployment: FALSE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 4.000
```

```
M-109
Class:  Land

Commodities: 4

155MM
StockTo 234.00 Deployment: TRUE
HighRate: 7.000 MedRate 4.970 LowRate 3.010 NoneRate 0.000

.50CAL
StockTo 0.12 Deployment: TRUE
HighRate: 0.017 MedRate 0.012 LowRate 0.007 NoneRate 0.000

DIESEL
StockTo 0.00 Deployment: FALSE
HighRate: 4.010 MedRate 4.010 LowRate 4.010 NoneRate 4.010

M-109
StockTo 1.00 Deployment: TRUE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
M-1A1
Class:  Land

Commodities: 5

120MM
StockTo 1.56 Deployment: TRUE
HighRate: 0.500 MedRate 0.360 LowRate 0.210 NoneRate 0.000

.50CAL
StockTo 0.03 Deployment: TRUE
HighRate: 0.004 MedRate 0.003 LowRate 0.002 NoneRate 0.000

5.56MM
StockTo 0.01 Deployment: TRUE
HighRate: 0.001 MedRate 0.001 LowRate 0.000 NoneRate 0.000

F44
StockTo 0.00 Deployment: FALSE
HighRate: 9.400 MedRate 9.400 LowRate 9.400 NoneRate 9.400

M-1A1
StockTo 1.00 Deployment: TRUE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

```
105HOWTWD
Class:  Land

Commodities: 2

105MM
StockTo 192.00 Deployment: TRUE
HighRate: 7.850 MedRate 5.580 LowRate 3.380 NoneRate 0.000

105HOW
StockTo 1.00 Deployment: TRUE
HighRate: 0.000 MedRate 0.000 LowRate 0.000 NoneRate 0.000
```

APPENDIX E

SWOS-129 UNIT FILES

```
MEF
Class:  Sea

Latitude:  23   0 N
Longitude:  63  0 E

DelayUntil: 0.00
InPlace: FALSE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 6 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 0


Commodities: 14

20MM/76
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 38.00 StockTo 38.00
OrderAt: 34.20 EmerOrderAt 34.20
Deployment: FALSE

F44
HighRate: 80.46 MedRate 52.00 LowRate 52.00 NoneRate 52.00
OnHand: 1856.00 StockTo 1856.00
OrderAt: 1484.80 EmerOrderAt 1484.80
Deployment: FALSE

F76
HighRate: 2879.00 MedRate 2879.00 LowRate 2879.00 NoneRate 2879.00
OnHand: 36350.00 StockTo 36350.00
OrderAt: 29080.00 EmerOrderAt 29080.00
Deployment: FALSE

SM-2MR
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 60.00 StockTo 60.00
OrderAt: 54.00 EmerOrderAt 54.00
Deployment: FALSE

HARPOON-S
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 16.00 StockTo 16.00
OrderAt: 14.40 EmerOrderAt 14.40
Deployment: FALSE

Mk-46
HighRate: 1.00 MedRate 0.50 LowRate 0.00 NoneRate 0.00
OnHand: 68.00 StockTo 68.00
OrderAt: 61.20 EmerOrderAt 61.20
Deployment: FALSE

TLAM-C
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
```

```
OnHand: 10.00 StockTo 10.00
OrderAt: 9.00 EmerOrderAt 9.00
Deployment: FALSE

TLAM-D
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

127MM/54
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 12.00 StockTo 12.00
OrderAt: 10.80 EmerOrderAt 10.80
Deployment: FALSE

SRBOC
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 40.00 StockTo 40.00
OrderAt: 36.00 EmerOrderAt 36.00
Deployment: FALSE

SM-1MR
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 72.00 StockTo 72.00
OrderAt: 64.80 EmerOrderAt 64.80
Deployment: FALSE

76MM/62
HighRate: 20.00 MedRate 10.00 LowRate 4.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 90.00
Deployment: FALSE

SONOBUOY
HighRate: 50.00 MedRate 50.00 LowRate 20.00 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

PENGUIN
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE
```

```
CTF50
Class:  Sea

Latitude:  24   0 N
Longitude:  64   0 E

DelayUntil: 0.00
InPlace: FALSE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 7 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 0


Commodities: 32

F44
HighRate: 8451.58 MedRate 6259.10 LowRate 6259.10 NoneRate 5709.30
OnHand: 62975.00 StockTo 62975.00
OrderAt: 50380.00 EmerOrderAt 50380.00
Deployment: FALSE

20MM
HighRate: 0.19 MedRate 0.19 LowRate 0.19 NoneRate 0.00
OnHand: 12.00 StockTo 12.00
OrderAt: 10.80 EmerOrderAt 10.80
Deployment: FALSE

RIM-7
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 40.00 StockTo 40.00
OrderAt: 36.00 EmerOrderAt 36.00
Deployment: FALSE

SRBOC
HighRate: 5.00 MedRate 2.00 LowRate 0.00 NoneRate 0.00
OnHand: 180.00 StockTo 180.00
OrderAt: 162.00 EmerOrderAt 162.00
Deployment: FALSE

HARPOON-A
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 30.00 StockTo 30.00
OrderAt: 27.00 EmerOrderAt 27.00
Deployment: FALSE

Mk-46
HighRate: 47.50 MedRate 24.75 LowRate 10.00 NoneRate 0.00
OnHand: 184.00 StockTo 184.00
OrderAt: 165.60 EmerOrderAt 165.60
Deployment: FALSE

SONOBUOY
HighRate: 955.00 MedRate 955.00 LowRate 470.00 NoneRate 0.00
```

```
OnHand: 3240.00 StockTo 3240.00
OrderAt: 2916.00 EmerOrderAt 2916.00
Deployment: FALSE

Mk-82
HighRate: 16.80 MedRate 16.80 LowRate 16.80 NoneRate 0.00
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 900.00
Deployment: FALSE

Mk-83
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 450.00
Deployment: FALSE

Mk-84
HighRate: 7.20 MedRate 7.20 LowRate 7.20 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 45.00
Deployment: FALSE

HARM
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 40.00 StockTo 40.00
OrderAt: 36.00 EmerOrderAt 36.00
Deployment: FALSE

AGM-62
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 40.00 StockTo 40.00
OrderAt: 36.00 EmerOrderAt 36.00
Deployment: FALSE

ROCKEYE
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 300.00 StockTo 300.00
OrderAt: 270.00 EmerOrderAt 270.00
Deployment: FALSE

AIM-7M
HighRate: 6.72 MedRate 6.72 LowRate 6.72 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 90.00
Deployment: FALSE

AIM-9M
HighRate: 5.52 MedRate 5.52 LowRate 2.40 NoneRate 0.00
OnHand: 150.00 StockTo 150.00
OrderAt: 135.00 EmerOrderAt 135.00
Deployment: FALSE

AIM-54C
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 45.00
Deployment: FALSE

TLAM-N
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
```

```
OnHand: 6.00 StockTo 6.00
OrderAt: 5.40 EmerOrderAt 5.40
Deployment: FALSE

TLAM-D
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 25.00 StockTo 25.00
OrderAt: 22.50 EmerOrderAt 22.50
Deployment: FALSE

TLAM-C
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 35.00 StockTo 35.00
OrderAt: 31.50 EmerOrderAt 31.50
Deployment: FALSE

TASM
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 25.00 StockTo 25.00
OrderAt: 22.50 EmerOrderAt 22.50
Deployment: FALSE

HARPOON-S
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 40.00 StockTo 40.00
OrderAt: 36.00 EmerOrderAt 36.00
Deployment: FALSE

SM-2MR
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 304.00 StockTo 304.00
OrderAt: 273.60 EmerOrderAt 273.60
Deployment: FALSE

ASROC
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 64.00 StockTo 64.00
OrderAt: 57.60 EmerOrderAt 57.60
Deployment: FALSE

127MM/54
HighRate: 30.00 MedRate 15.00 LowRate 3.00 NoneRate 0.00
OnHand: 124.00 StockTo 124.00
OrderAt: 111.60 EmerOrderAt 111.60
Deployment: FALSE

20MM/76
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 45.00
Deployment: FALSE

F76
HighRate: 4050.00 MedRate 4050.00 LowRate 4050.00 NoneRate 4050.00
OnHand: 60950.00 StockTo 60950.00
OrderAt: 48760.00 EmerOrderAt 48760.00
Deployment: FALSE

SM-1MR
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
```

```
OnHand: 72.00 StockTo 72.00
OrderAt: 64.80 EmerOrderAt 64.80
Deployment: FALSE

76MM/62
HighRate: 20.00 MedRate 10.00 LowRate 4.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 90.00
Deployment: FALSE

AGM-65
HighRate: 13.20 MedRate 13.20 LowRate 13.20 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

Mines
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

DEPTHCHARGE
HighRate: 16.00 MedRate 8.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

PENGUIN
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE
```

```
CTG51.1A
Class:  Sea

Latitude:  23   0 N
Longitude:  63   0 E

DelayUntil: 0.00
InPlace: TRUE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 3 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 0


Commodities: 15

RIM-7
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 16.00 StockTo 16.00
OrderAt: 14.40 EmerOrderAt 14.40
Deployment: FALSE

20MM/76
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 38.00 StockTo 38.00
OrderAt: 34.20 EmerOrderAt 34.20
Deployment: FALSE

SRBOC
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 160.00 StockTo 160.00
OrderAt: 144.00 EmerOrderAt 144.00
Deployment: FALSE

Mk-82
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 90.00
Deployment: FALSE

AIM-9L
HighRate: 2.30 MedRate 1.78 LowRate 1.78 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 45.00
Deployment: FALSE

AGM-65
HighRate: 0.60 MedRate 0.60 LowRate 0.60 NoneRate 0.00
OnHand: 20.00 StockTo 20.00
OrderAt: 18.00 EmerOrderAt 18.00
Deployment: FALSE

AGM-62
HighRate: 0.60 MedRate 0.60 LowRate 0.60 NoneRate 0.00
```

```
OnHand: 20.00 StockTo 20.00
OrderAt: 18.00 EmerOrderAt 18.00
Deployment: FALSE

PAVEWAY
HighRate: 3.00 MedRate 3.00 LowRate 3.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 90.00
Deployment: FALSE

F44
HighRate: 1588.45 MedRate 1563.88 LowRate 977.06 NoneRate 977.06
OnHand: 31000.00 StockTo 31000.00
OrderAt: 24800.00 EmerOrderAt 24800.00
Deployment: FALSE

F76
HighRate: 3000.00 MedRate 3000.00 LowRate 3000.00 NoneRate 3000.00
OnHand: 42855.00 StockTo 42855.00
OrderAt: 34284.00 EmerOrderAt 34284.00
Deployment: FALSE

25MM
HighRate: 0.04 MedRate 0.04 LowRate 0.04 NoneRate 0.00
OnHand: 4.00 StockTo 4.00
OrderAt: 3.60 EmerOrderAt 3.60
Deployment: FALSE

TOW
HighRate: 32.00 MedRate 32.00 LowRate 8.00 NoneRate 0.00
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 900.00
Deployment: FALSE

HELLFIRE
HighRate: 16.00 MedRate 16.00 LowRate 8.00 NoneRate 0.00
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 450.00
Deployment: FALSE

20MM
HighRate: 0.04 MedRate 0.04 LowRate 0.04 NoneRate 0.00
OnHand: 2.00 StockTo 2.00
OrderAt: 1.80 EmerOrderAt 1.80
Deployment: FALSE

40MMGREN
HighRate: 0.01 MedRate 0.01 LowRate 0.01 NoneRate 0.00
OnHand: 1.00 StockTo 1.00
OrderAt: 0.90 EmerOrderAt 0.90
Deployment: FALSE
```

```
CTG51.1B
Class:  Sea

Latitude:  23   0 N
Longitude:  63   0 E

DelayUntil: 0.00
InPlace: TRUE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 3 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 0


Commodities: 15

127MM/54
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 12.00 StockTo 12.00
OrderAt: 10.80 EmerOrderAt 10.80
Deployment: FALSE

20MM/76
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 48.00 StockTo 48.00
OrderAt: 43.20 EmerOrderAt 43.20
Deployment: FALSE

SRBOC
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 120.00 StockTo 120.00
OrderAt: 108.00 EmerOrderAt 108.00
Deployment: FALSE

25MM
HighRate: 0.04 MedRate 0.04 LowRate 0.04 NoneRate 0.00
OnHand: 4.00 StockTo 4.00
OrderAt: 3.60 EmerOrderAt 3.60
Deployment: FALSE

AIM-9L
HighRate: 2.30 MedRate 1.78 LowRate 1.78 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 45.00
Deployment: FALSE

AGM-65
HighRate: 0.60 MedRate 0.60 LowRate 0.60 NoneRate 0.00
OnHand: 20.00 StockTo 20.00
OrderAt: 18.00 EmerOrderAt 18.00
Deployment: FALSE

AGM-62
HighRate: 0.60 MedRate 0.60 LowRate 0.60 NoneRate 0.00
```

```
OnHand: 20.00 StockTo 20.00
OrderAt: 18.00 EmerOrderAt 18.00
Deployment: FALSE

PAVEWAY
HighRate: 3.00 MedRate 3.00 LowRate 3.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 90.00
Deployment: FALSE

F44
HighRate: 1588.45 MedRate 1563.88 LowRate 977.06 NoneRate 977.06
OnHand: 31000.00 StockTo 31000.00
OrderAt: 24800.00 EmerOrderAt 24800.00
Deployment: FALSE

F76
HighRate: 3000.00 MedRate 3000.00 LowRate 3000.00 NoneRate 3000.00
OnHand: 42855.00 StockTo 42855.00
OrderAt: 34284.00 EmerOrderAt 34284.00
Deployment: FALSE

Mk-82
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 90.00
Deployment: FALSE

TOW
HighRate: 32.00 MedRate 32.00 LowRate 8.00 NoneRate 0.00
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 900.00
Deployment: FALSE

HELLFIRE
HighRate: 16.00 MedRate 16.00 LowRate 8.00 NoneRate 0.00
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 450.00
Deployment: FALSE

20MM
HighRate: 0.04 MedRate 0.04 LowRate 0.04 NoneRate 0.00
OnHand: 2.00 StockTo 2.00
OrderAt: 1.80 EmerOrderAt 1.80
Deployment: FALSE

40MMGREN
HighRate: 0.01 MedRate 0.01 LowRate 0.01 NoneRate 0.00
OnHand: 1.00 StockTo 1.00
OrderAt: 0.90 EmerOrderAt 0.90
Deployment: FALSE
```

```
CTF51.2
Class:  Sea

Latitude:  23   0 N
Longitude:  63  0 E

DelayUntil: 0.00
InPlace: FALSE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 2 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 1

AE27 1

Commodities: 17

TLAM-N
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 2.00 StockTo 2.00
OrderAt: 1.80 EmerOrderAt 1.80
Deployment: FALSE

TLAM-D
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 5.00 StockTo 5.00
OrderAt: 4.50 EmerOrderAt 4.50
Deployment: FALSE

TLAM-C
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 5.00 StockTo 5.00
OrderAt: 4.50 EmerOrderAt 4.50
Deployment: FALSE

TASM
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 5.00 StockTo 5.00
OrderAt: 4.50 EmerOrderAt 4.50
Deployment: FALSE

HARPOON-S
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 12.00 StockTo 12.00
OrderAt: 10.80 EmerOrderAt 10.80
Deployment: FALSE

RIM-7
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 24.00 StockTo 24.00
OrderAt: 21.60 EmerOrderAt 21.60
Deployment: FALSE

ASROC
```

```
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 24.00 StockTo 24.00
OrderAt: 21.60 EmerOrderAt 21.60
Deployment: FALSE

127MM/54
HighRate: 10.00 MedRate 5.00 LowRate 1.00 NoneRate 0.00
OnHand: 12.00 StockTo 12.00
OrderAt: 10.80 EmerOrderAt 10.80
Deployment: FALSE

20MM/76
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 19.00 StockTo 19.00
OrderAt: 17.10 EmerOrderAt 17.10
Deployment: FALSE

Mk-46
HighRate: 1.00 MedRate 0.50 LowRate 0.00 NoneRate 0.00
OnHand: 38.00 StockTo 38.00
OrderAt: 34.20 EmerOrderAt 34.20
Deployment: FALSE

SRBOC
HighRate: 5.00 MedRate 2.00 LowRate 0.00 NoneRate 0.00
OnHand: 40.00 StockTo 40.00
OrderAt: 36.00 EmerOrderAt 36.00
Deployment: FALSE

F76
HighRate: 1311.00 MedRate 1311.00 LowRate 1311.00 NoneRate 1311.00
OnHand: 16190.00 StockTo 16190.00
OrderAt: 12952.00 EmerOrderAt 12952.00
Deployment: FALSE

F44
HighRate: 80.46 MedRate 52.00 LowRate 52.00 NoneRate 52.00
OnHand: 952.00 StockTo 952.00
OrderAt: 761.60 EmerOrderAt 761.60
Deployment: FALSE

SONOBUOY
HighRate: 50.00 MedRate 50.00 LowRate 20.00 NoneRate 0.00
OnHand: 230.00 StockTo 230.00
OrderAt: 207.00 EmerOrderAt 207.00
Deployment: FALSE

SM-1MR
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 36.00 StockTo 36.00
OrderAt: 32.40 EmerOrderAt 32.40
Deployment: FALSE

76MM/62
HighRate: 10.00 MedRate 5.00 LowRate 2.00 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 45.00
Deployment: FALSE

PENGUIN
```

```
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE
```

```
AF-MASIRAH
Class:  Air

Latitude:  20  30 N
Longitude:  58 30 E

DelayUntil: 0.00
InPlace: TRUE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 0


Commodities: 8

20MM
HighRate: 0.14 MedRate 0.14 LowRate 0.14 NoneRate 0.00
OnHand: 30.00 StockTo 30.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

AIM-7M
HighRate: 1.68 MedRate 1.68 LowRate 1.68 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

AIM-9M
HighRate: 1.38 MedRate 1.38 LowRate 1.38 NoneRate 0.00
OnHand: 150.00 StockTo 150.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

F44
HighRate: 3681.90 MedRate 2086.38 LowRate 2086.38 NoneRate 2086.38
OnHand: 110000.00 StockTo 110000.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

Mk-82
HighRate: 26.40 MedRate 26.40 LowRate 26.40 NoneRate 0.00
OnHand: 1000.00 StockTo 1000.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

AGM-65
HighRate: 1.20 MedRate 1.20 LowRate 1.20 NoneRate 0.00
OnHand: 40.00 StockTo 40.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

ROCKEYE
HighRate: 7.20 MedRate 7.20 LowRate 7.20 NoneRate 0.00
```

```
OnHand: 220.00 StockTo 220.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE

PAVEWAY
HighRate: 6.00 MedRate 6.00 LowRate 6.00 NoneRate 0.00
OnHand: 180.00 StockTo 180.00
OrderAt: 0.00 EmerOrderAt 0.00
Deployment: FALSE
```

```
AF-RIYADH
Class:  Air

Latitude:  25   0 N
Longitude:  47  0 E

DelayUntil: 0.00
InPlace: TRUE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: TRUE  MaxCapacity: 20 MaxSize: 90000.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 50 MaxSize: 10.00

TransporterTypes: 0


Commodities: 1

F44
HighRate: 4090.90 MedRate 4090.90 LowRate 3068.20 NoneRate 3068.20
OnHand: 400000.00 StockTo 400000.00
OrderAt: 320000.00 EmerOrderAt 240000.00
Deployment: FALSE
```

```
VP-1
Class:  Air

Latitude:   3   0 N
Longitude:  72 30 E

DelayUntil: 0.00
InPlace: TRUE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 0


Commodities: 5

HARPOON-A
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 30.00 StockTo 30.00
OrderAt: 25.00 EmerOrderAt 0.00
Deployment: FALSE

Mk-46
HighRate: 16.00 MedRate 8.00 LowRate 0.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00
Deployment: FALSE

SONOBUOY
HighRate: 200.00 MedRate 100.00 LowRate 50.00 NoneRate 0.00
OnHand: 2000.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00
Deployment: FALSE

Mines
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 0.00
Deployment: FALSE

F44
HighRate: 2209.12 MedRate 2209.12 LowRate 1656.80 NoneRate 1656.80
OnHand: 66000.00 StockTo 66000.00
OrderAt: 45000.00 EmerOrderAt 0.00
Deployment: FALSE
```

```
VP-8
Class:  Air

Latitude:  20  30 N
Longitude:  58 30 E

DelayUntil: 0.00
InPlace: TRUE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 0


Commodities: 5

HARPOON-A
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 30.00 StockTo 30.00
OrderAt: 25.00 EmerOrderAt 0.00
Deployment: FALSE

Mk-46
HighRate: 16.00 MedRate 8.00 LowRate 0.00 NoneRate 0.00
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00
Deployment: FALSE

SONOBUOY
HighRate: 200.00 MedRate 100.00 LowRate 50.00 NoneRate 0.00
OnHand: 2000.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00
Deployment: FALSE

Mines
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 50.00 StockTo 50.00
OrderAt: 45.00 EmerOrderAt 0.00
Deployment: FALSE

F44
HighRate: 2209.12 MedRate 2209.12 LowRate 1656.80 NoneRate 1656.80
OnHand: 66000.00 StockTo 66000.00
OrderAt: 45000.00 EmerOrderAt 0.00
Deployment: FALSE
```

```
VQ-1
Class:  Air

Latitude:  20  30 n
Longitude:  58 30 e

DelayUntil: 0.00
InPlace: TRUE
ActiveAt: 0.00
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 1

TANKTRUCKS 1

Commodities: 1

F44
HighRate: 414.20 MedRate 414.20 LowRate 414.20 NoneRate 414.20
OnHand: 12000.00 StockTo 12000.00
OrderAt: 9600.00 EmerOrderAt 7200.00
Deployment: FALSE
```

```
3ACR
Class:  Land

Latitude:  25   0 N
Longitude:  67   0 E

DelayUntil: 100.00
InPlace: FALSE
ActiveAt: 0.90
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 20.00

TransporterTypes: 2

TANKTRUCKS 5
TRUCKCONVOY 10

Commodities: 7

Personnel
HighRate: 47.00 MedRate 27.00 LowRate 18.00 NoneRate 3.00
OnHand: 0.00 StockTo 4663.00
OrderAt: 4563.00 EmerOrderAt 2300.00
Deployment: TRUE

F44
HighRate: 1234.27 MedRate 1154.98 LowRate 1154.98 NoneRate 1075.69
OnHand: 0.00 StockTo 37020.00
OrderAt: 18510.00 EmerOrderAt 122212.00
Deployment: FALSE

DIESEL
HighRate: 72.18 MedRate 72.18 LowRate 72.18 NoneRate 72.18
OnHand: 0.00 StockTo 2165.40
OrderAt: 1732.32 EmerOrderAt 1299.00
Deployment: FALSE

M-1A1
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 116.00
OrderAt: 115.00 EmerOrderAt 58.00
Deployment: TRUE

AH-64
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 9.00
OrderAt: 8.00 EmerOrderAt 5.00
Deployment: TRUE

UNITEQ
HighRate: 1029.00 MedRate 1300.00 LowRate 900.00 NoneRate 0.00
OnHand: 0.00 StockTo 30762.00
OrderAt: 24609.00 EmerOrderAt 15500.00
Deployment: TRUE
```

```
MOGAS
HighRate: 178.00 MedRate 178.00 LowRate 178.00 NoneRate 178.00
OnHand: 0.00 StockTo 5350.00
OrderAt: 4280.00 EmerOrderAt 3210.00
Deployment: FALSE
```

```
82ABNDIV
Class:  Land

Latitude:  25   0 n
Longitude:  67  0 e

DelayUntil: 100.00
InPlace: FALSE
ActiveAt: 0.90
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 20.00

TransporterTypes: 2

TRUCKCONVOY 10
TANKTRUCKS 5

Commodities: 7

Personnel
HighRate: 301.00 MedRate 172.00 LowRate 117.00 NoneRate 3.00
OnHand: 0.00 StockTo 11674.00
OrderAt: 11574.00 EmerOrderAt 6000.00
Deployment: TRUE

AH-64
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 33.00
OrderAt: 32.00 EmerOrderAt 25.00
Deployment: TRUE

105HOW
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 0.00 StockTo 54.00
OrderAt: 53.00 EmerOrderAt 27.00
Deployment: TRUE

UNITEQ
HighRate: 1555.00 MedRate 1781.00 LowRate 1400.00 NoneRate 0.00
OnHand: 0.00 StockTo 21942.00
OrderAt: 17553.60 EmerOrderAt 11000.00
Deployment: TRUE

F44
HighRate: 1811.00 MedRate 1811.00 LowRate 1811.00 NoneRate 1811.00
OnHand: 0.00 StockTo 54340.00
OrderAt: 43472.00 EmerOrderAt 32604.00
Deployment: FALSE

DIESEL
HighRate: 536.50 MedRate 536.50 LowRate 536.00 NoneRate 536.00
OnHand: 0.00 StockTo 16095.00
OrderAt: 12876.00 EmerOrderAt 9657.00
Deployment: FALSE
```

```
MOGAS
HighRate: 579.00 MedRate 579.00 LowRate 579.00 NoneRate 579.00
OnHand: 0.00 StockTo 17396.00
OrderAt: 13916.00 EmerOrderAt 10437.00
Deployment: FALSE
```

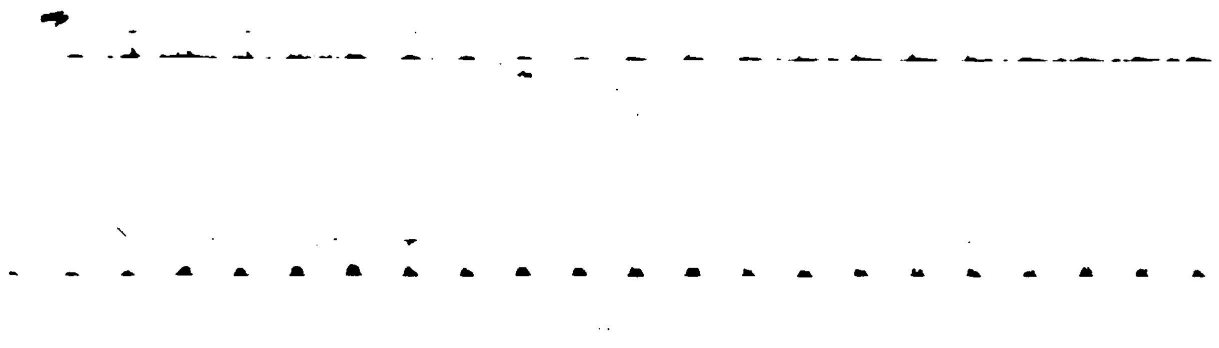
```
7MEB
Class:  Land

Latitude:  25   0 n
Longitude:  67  0 e

DelayUntil: 100.00
InPlace: FALSE
ActiveAt: 0.90
InitialCombatIntensity: None

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 20.00

TransporterTypes: 1

TRUCKCONVOY 4

Commodities: 6

Personnel
HighRate: 301.00 MedRate 172.00 LowRate 117.00 NoneRate 3.00
OnHand: 4513.00 StockTo 4513.00
OrderAt: 4413.00 EmerOrderAt 2250.00
Deployment: TRUE

105HOW
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 4.00 StockTo 4.00
OrderAt: 3.00 EmerOrderAt 2.00
Deployment: TRUE

M-109
HighRate: 0.00 MedRate 0.00 LowRate 0.00 NoneRate 0.00
OnHand: 12.00 StockTo 12.00
OrderAt: 11.00 EmerOrderAt 6.00
Deployment: TRUE

UNITEQ
HighRate: 1029.00 MedRate 1300.00 LowRate 900.00 NoneRate 0.00
OnHand: 7737.00 StockTo 7737.50
OrderAt: 6189.00 EmerOrderAt 3868.00
Deployment: TRUE

DIESEL
HighRate: 1530.00 MedRate 1530.00 LowRate 1530.00 NoneRate 1530.00
OnHand: 0.00 StockTo 45900.00
OrderAt: 36720.00 EmerOrderAt 27540.00
Deployment: FALSE

MOGAS
HighRate: 117.00 MedRate 117.00 LowRate 117.00 NoneRate 117.00
OnHand: 0.00 StockTo 3500.00
OrderAt: 2800.00 EmerOrderAt 2320.00
Deployment: FALSE
```

APPENDIX F

SWOS-129 BASE FILES

```
Guam

Group:  ILOC SubGroup:  NONE

Latitude:  14   0 n
Longitude: 145  0 e

HasAirport: TRUE  MaxCapacity: 20 MaxSize: 60000.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1000.00
HasRail: FALSE  MaxCapacity: 1 MaxSize: 10.00
HasTruckStop: TRUE  MaxCapacity: 2 MaxSize: 20.00

TransporterTypes: 2

C-141B 5
C-5A 2

Commodities: 10

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 777000.00 StockTo 777000.00
OrderAt: 6216000.00 EmerOrderAt 310500.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 343000.00 StockTo 343000.00
OrderAt: 257250.00 EmerOrderAt 171500.00

DIESEL
OnHand: 343000.00 StockTo 343000.00
OrderAt: 257250.00 EmerOrderAt 171500.00

MOGAS
OnHand: 343000.00 StockTo 343000.00
OrderAt: 257250.00 EmerOrderAt 171500.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
UNITEQ
OnHand: 0.00 StockTo 7737.50
OrderAt: 6189.00 EmerOrderAt 3868.00
```



```
PINTOINT

Group:  THEATER SubGroup:  POD

Latitude:  25   0 N
Longitude:  67  0 E

HasAirport: FALSE  MaxCapacity: 20 MaxSize: 90000.00
HasSeaport: FALSE  MaxCapacity: 10 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 50 MaxSize: 10.00

TransporterTypes: 0


Commodities: 1

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
DIEGO

Group:  ILOC SubGroup:  POS

Latitude:   3   0 N
Longitude:  72 30 E

HasAirport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 4

A0177 1
AE27 1
TRUCKCONVOY 5
TANKTRUCKS 1

Commodities: 49

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 420000.00 StockTo 420000.00
OrderAt: 336000.00 EmerOrderAt 210000.00

MOGAS
OnHand: 12000.00 StockTo 12000.00
OrderAt: 10000.00 EmerOrderAt 6000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 198000.00 StockTo 198000.00
OrderAt: 154400.00 EmerOrderAt 99000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Mk-82
OnHand: 2000.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00

AIM-9M
OnHand: 2000.00 StockTo 2000.00
```

```
OrderAt: 1800.00 EmerOrderAt 0.00

AIM-7M
OnHand: 2000.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00

AIM-54C
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

Mines
OnHand: 200.00 StockTo 200.00
OrderAt: 150.00 EmerOrderAt 0.00

20MM
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

20MM/76
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

RIM-7
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

SRBOC
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

TLAM-D
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

TLAM-C
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

TASM
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

HARPOON-S
OnHand: 200.00 StockTo 200.00
OrderAt: 180.00 EmerOrderAt 0.00

HARPOON-A
OnHand: 200.00 StockTo 200.00
OrderAt: 180.00 EmerOrderAt 0.00

ASROC
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

Mk-46
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

SM-2ER
OnHand: 100.00 StockTo 100.00
```

```
OrderAt: 90.00 EmerOrderAt 0.00

SM-2MR
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

127MM/54
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

76MM/62
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

25MM
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

76MM/50
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

HARM
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

AMRAAM
OnHand: 300.00 StockTo 300.00
OrderAt: 270.00 EmerOrderAt 0.00

AGM-62
OnHand: 800.00 StockTo 800.00
OrderAt: 720.00 EmerOrderAt 0.00

AGM-65
OnHand: 800.00 StockTo 800.00
OrderAt: 720.00 EmerOrderAt 0.00

SONOBUOY
OnHand: 5000.00 StockTo 5000.00
OrderAt: 4500.00 EmerOrderAt 0.00

DEPTHCHARGE
OnHand: 300.00 StockTo 300.00
OrderAt: 270.00 EmerOrderAt 0.00

PAVEWAY
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

AIM-9L
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

TOW
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

HELLFIRE
OnHand: 500.00 StockTo 500.00
```

```
OrderAt: 450.00 EmerOrderAt 0.00

40MMGREN
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

Mk-83
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

Mk-84
OnHand: 250.00 StockTo 250.00
OrderAt: 225.00 EmerOrderAt 0.00

ROCKEYE
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

FAE
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

155MM
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

105MM
OnHand: 750.00 StockTo 750.00
OrderAt: 675.00 EmerOrderAt 0.00

81MM
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

SM-1MR
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00
```

```
PINTOINT

Group:  THEATER SubGroup:  POD

Latitude:  25   0 N
Longitude:  67  0 E

HasAirport: FALSE  MaxCapacity: 20 MaxSize: 90000:00
HasSeaport: FALSE  MaxCapacity: 10 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 50 MaxSize: 10.00

TransporterTypes: 0


Commodities: 1

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

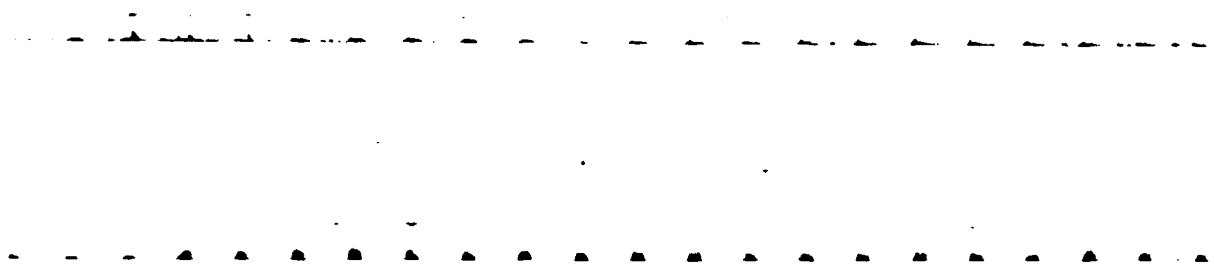
```
MASIRAH

Group:  THEATER SubGroup:  POS

Latitude:  20  30 N
Longitude:  58 30 E

HasAirport: TRUE  MaxCapacity: 20 MaxSize: 90000.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 7

C-5A 5
C-141B 10
AE27 1
AO177 1
TANKER 1
TRUCKCONVOY 5
TANKTRUCKS 2

Commodities: 0
```

```
NWSEarle

Group:  CONUS SubGroup:  NONE

Latitude:  40  40 n
Longitude:  74 10 w

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 800.00
HasRail: TRUE  MaxCapacity: 10 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 100 MaxSize: 10.00

TransporterTypes: 3

Breakbulk 5
UNITTRAIN 1
FREIGHTTRAIN 1

Commodities: 41

Mk-82
OnHand: 2000.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00

AIM-9M
OnHand: 2000.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00

AIM-7M
OnHand: 2000.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00

AIM-54C
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

Mines
OnHand: 200.00 StockTo 200.00
OrderAt: 150.00 EmerOrderAt 0.00

20MM
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

20MM/76
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

RIM-7
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

SRBOC
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

TLAM-D
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00
```

```
TLAM-C
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

TASM
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

HARPOON-S
OnHand: 200.00 StockTo 200.00
OrderAt: 180.00 EmerOrderAt 0.00

HARPOON-A
OnHand: 200.00 StockTo 200.00
OrderAt: 180.00 EmerOrderAt 0.00

ASROC
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

Mk-46
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

SM-2ER
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

SM-2MR
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

127MM/54
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

76MM/62
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

25MM
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

76MM/50
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

HARM
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

AMRAAM
OnHand: 300.00 StockTo 300.00
OrderAt: 270.00 EmerOrderAt 0.00

AGM-62
OnHand: 800.00 StockTo 800.00
OrderAt: 720.00 EmerOrderAt 0.00
```

```
AGM-65
OnHand: 800.00 StockTo 800.00
OrderAt: 720.00 EmerOrderAt 0.00

SONOBUOY
OnHand: 5000.00 StockTo 5000.00
OrderAt: 4500.00 EmerOrderAt 0.00

DEPTHCHARGE
OnHand: 300.00 StockTo 300.00
OrderAt: 270.00 EmerOrderAt 0.00

PAVEWAY
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

AIM-9L
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

TOW
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00

HELLFIRE
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

40MMGREN
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00

Mk-83
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

Mk-84
OnHand: 250.00 StockTo 250.00
OrderAt: 225.00 EmerOrderAt 0.00

ROCKEYE
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

FAE
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

155MM
OnHand: 500.00 StockTo 500.00
OrderAt: 450.00 EmerOrderAt 0.00

105MM
OnHand: 750.00 StockTo 750.00
OrderAt: 675.00 EmerOrderAt 0.00

81MM
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 0.00
```

```
SM-1MR
OnHand: 100.00 StockTo 100.00
OrderAt: 90.00 EmerOrderAt 0.00
```



```
MOTBayonne

Group:  CONUS SubGroup:  POE

Latitude:  40  40 n
Longitude:  74 10 w

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 10 MaxSize: 1000.00
HasRail: TRUE  MaxCapacity: 25 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 100 MaxSize: 10.00

TransporterTypes: 4

C4-S-1H 3
FREIGHTTRAIN 3
UNITTRAIN 3
TANKERTRAIN 3

Commodities: 0
```

```
NSCNorfolk

Group:  CONUS SubGroup:  POE

Latitude:  36  50 n
Longitude:  76 20 w

HasAirport: TRUE  MaxCapacity: 50 MaxSize: 90000.00
HasSeaport: TRUE  MaxCapacity: 12 MaxSize: 1000.00
HasRail: TRUE  MaxCapacity: 1 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 3

FREIGHTTRAIN 1
TANKERTRAIN 1
PAXTRAIN 3

Commodities: 8

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 1000000.00 StockTo 1000000.00
OrderAt: 800000.00 EmerOrderAt 250000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 2000000.00 StockTo 2000000.00
OrderAt: 1800000.00 EmerOrderAt 500000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
MOTOakland

Group:  CONUS SubGroup:  POE

Latitude:  37  50 n
Longitude: 122 20 w

HasAirport: TRUE  MaxCapacity: 20 MaxSize: 275000.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1000.00
HasRail: TRUE  MaxCapacity: 25 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 100 MaxSize: 10.00

TransporterTypes: 2

Breakbulk 5
TANKER 2

Commodities: 8

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 1000000.00 StockTo 1000000.00
OrderAt: 800000.00 EmerOrderAt 250000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 2000000.00 StockTo 2000000.00
OrderAt: 1800000.00 EmerOrderAt 500000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
NSCSanDiego

Group:  CONUS SubGroup:  NONE

Latitude:  32  40 n
Longitude: 117 10 w

HasAirport: TRUE  MaxCapacity: 40 MaxSize: 275000.00
HasSeaport: TRUE  MaxCapacity: 2 MaxSize: 1000.00
HasRail: TRUE  MaxCapacity: 25 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 100 MaxSize: 10.00

TransporterTypes: 6

C-141B 5
TANKER 2
C4-S-1H 4
FREIGHTTRAIN 2
TANKERTRAIN 1
PAXTRAIN 1

Commodities: 8

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 1000000.00 StockTo 1000000.00
OrderAt: 800000.00 EmerOrderAt 250000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 2000000.00 StockTo 2000000.00
OrderAt: 1800000.00 EmerOrderAt 500000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
Singapore

Group:  ILOC SubGroup:  NONE

Latitude:   2  30 n
Longitude: 104  0 e

HasAirport: TRUE  MaxCapacity: 20 MaxSize: 275000.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1000.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 2

AO177 1
AE27 1

Commodities: 8

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 154000.00 StockTo 154000.00
OrderAt: 1232000.00 EmerOrderAt 77000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 371000.00 StockTo 371000.00
OrderAt: 296800.00 EmerOrderAt 185500.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
Rota

Group:  ILOC SubGroup:  NONE

Latitude:  36   0 n
Longitude:   5  0 w

HasAirport: TRUE  MaxCapacity: 20 MaxSize: 275000.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1000.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 2

TANKER 2
AE27 2

Commodities: 8

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 140000.00 StockTo 140000.00
OrderAt: 112000.00 EmerOrderAt 70000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 288000.00 EmerOrderAt 10000.00

F76
OnHand: 360000.00 StockTo 360000.00
OrderAt: 1800000.00 EmerOrderAt 180000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
Sigonella

Group:  ILOC SubGroup:  NONE

Latitude:  37  30 n
Longitude:  14 30 e

HasAirport: TRUE  MaxCapacity: 20 MaxSize: 275000.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1000.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 0


Commodities: 8

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 1000000.00 StockTo 1000000.00
OrderAt: 800000.00 EmerOrderAt 250000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 2000000.00 StockTo 2000000.00
OrderAt: 1800000.00 EmerOrderAt 500000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
Okinawa

Group:  ILOC SubGroup:  NONE

Latitude:  26   0 n
Longitude: 128  0 e

HasAirport: TRUE  MaxCapacity: 50 MaxSize: 275000.00
HasSeaport: TRUE  MaxCapacity: 2 MaxSize: 800.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 0 MaxSize: 0.00

TransporterTypes: 2

C-5A 10
C-141B 15

Commodities: 8

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 1000000.00 StockTo 1000000.00
OrderAt: 800000.00 EmerOrderAt 250000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 2000000.00 StockTo 2000000.00
OrderAt: 1800000.00 EmerOrderAt 500000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
FtBragg

Group:  CONUS SubGroup:  NONE

Latitude:  35  10 n
Longitude:  79  1 w

HasAirport: FALSE  MaxCapacity: 60 MaxSize: 275000.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: TRUE  MaxCapacity: 25 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 100 MaxSize: 10.00

TransporterTypes: 3

FREIGHTTRAIN 2
UNITTRAIN 4
PAXTRAIN 4

Commodities: 4

Personnel
OnHand: 11674.00 StockTo 2000.00
OrderAt: 1800.00 EmerOrderAt 0.00

AH-64
OnHand: 33.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00

105HOW
OnHand: 54.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00

UNITEQ
OnHand: 21942.00 StockTo 21942.00
OrderAt: 0.00 EmerOrderAt 0.00
```

```
PopeAFB

Group:  CONUS SubGroup:  POE

Latitude:  35  10 n
Longitude:  79  2 w

HasAirport: TRUE  MaxCapacity: 40 MaxSize: 275000.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: TRUE  MaxCapacity: 4 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 6

C-5A 5
C-141B 10
B747 5
FREIGHTTRAIN 1
TANKERTRAIN 1
PAXTRAIN 3

Commodities: 0
```

```
Wilmington

Group:  CONUS SubGroup:  POE

Latitude:  34  15 n
Longitude:  78  0 w

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 18 MaxSize: 1000.00
HasRail: TRUE  MaxCapacity: 4 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 3

FREIGHTTRAIN 1
TANKERTRAIN 1
PAXTRAIN 3

Commodities: 0
```

```
FtBliss

Group:  CONUS SubGroup:  NONE

Latitude:  31  45 n
Longitude: 106 30 w

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: TRUE  MaxCapacity: 25 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 100 MaxSize: 10.00

TransporterTypes: 3

FREIGHTTRAIN 2
UNITTRAIN 4
PAXTRAIN 4

Commodities: 6

Personnel
OnHand: 4663.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00

F44
OnHand: 37020.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00

DIESEL
OnHand: 2165.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00

M-1A1
OnHand: 116.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00

AH-64
OnHand: 9.00 StockTo 0.00
OrderAt: 0.00 EmerOrderAt 0.00

UNITEQ
OnHand: 30762.00 StockTo 30762.00
OrderAt: 0.00 EmerOrderAt 0.00
```

```
HollomanAFB

Group:  CONUS SubGroup:  POE

Latitude:  33   0 n
Longitude: 106  5 w

HasAirport: TRUE  MaxCapacity: 40 MaxSize: 275000.00
HasSeaport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasRail: TRUE  MaxCapacity: 4 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 6

C-5A 5
C-141B 10
B747 5
FREIGHTTRAIN 1
TANKERTRAIN 1
PAXTRAIN 3

Commodities: 0
```

```
Corpus

Group:  CONUS SubGroup:  POE

Latitude:  27  45 n
Longitude:  97 20 w

HasAirport: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasSeaport: TRUE  MaxCapacity: 21 MaxSize: 1000.00
HasRail: TRUE  MaxCapacity: 4 MaxSize: 40.00
HasTruckStop: TRUE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 3

FREIGHTTRAIN 1
TANKERTRAIN 1
PAXTRAIN 3

Commodities: 0
```

```
Djibouti

Group:  ILOC SubGroup:  POS

Latitude:  43   0 N
Longitude:  11 50 E

HasAirport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 1

TANKER 2

Commodities: 10

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 180000.00 StockTo 180000.00
OrderAt: 144000.00 EmerOrderAt 210000.00

MOGAS
OnHand: 12000.00 StockTo 12000.00
OrderAt: 10000.00 EmerOrderAt 6000.00

DIESEL
OnHand: 200000.00 StockTo 200000.00
OrderAt: 160000.00 EmerOrderAt 120000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 187000.00 StockTo 187000.00
OrderAt: 149600.00 EmerOrderAt 99000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
Muscat

Group:  ILOC SubGroup:  POS

Latitude:  24   0 N
Longitude:  58   0 E

HasAirport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 3

A0177 1
AE27 1
TANKER 2

Commodities: 10

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 118000.00 StockTo 118000.00
OrderAt: 94400.00 EmerOrderAt 70800.00

MOGAS
OnHand: 12000.00 StockTo 12000.00
OrderAt: 10000.00 EmerOrderAt 6000.00

DIESEL
OnHand: 200000.00 StockTo 200000.00
OrderAt: 160000.00 EmerOrderAt 120000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 251000.00 StockTo 251000.00
OrderAt: 200800.00 EmerOrderAt 150600.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
Bahrain

Group:  ILOC SubGroup:  POS

Latitude:  26  30 N
Longitude:  51  0 E

HasAirport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 3

A0177 1
AE27 1
TANKER 2

Commodities: 9

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 1600000.00 StockTo 1600000.00
OrderAt: 1280000.00 EmerOrderAt 960000.00

MOGAS
OnHand: 12000.00 StockTo 12000.00
OrderAt: 10000.00 EmerOrderAt 6000.00

DIESEL
OnHand: 200000.00 StockTo 200000.00
OrderAt: 160000.00 EmerOrderAt 120000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

```
Dubai

Group:  ILOC SubGroup:  POS

Latitude:  25   0 N
Longitude:  55 30 E

HasAirport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasSeaport: TRUE  MaxCapacity: 5 MaxSize: 1500.00
HasRail: FALSE  MaxCapacity: 0 MaxSize: 0.00
HasTruckStop: FALSE  MaxCapacity: 20 MaxSize: 10.00

TransporterTypes: 3

AO177 1
AE27 1
TANKER 2

Commodities: 10

Personnel
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

F44
OnHand: 756000.00 StockTo 756000.00
OrderAt: 604800.00 EmerOrderAt 300000.00

MOGAS
OnHand: 12000.00 StockTo 12000.00
OrderAt: 10000.00 EmerOrderAt 6000.00

DIESEL
OnHand: 200000.00 StockTo 200000.00
OrderAt: 160000.00 EmerOrderAt 120000.00

General
OnHand: 40000.00 StockTo 40000.00
OrderAt: 36000.00 EmerOrderAt 10000.00

F76
OnHand: 201000.00 StockTo 201000.00
OrderAt: 164400.00 EmerOrderAt 100000.00

FFV
OnHand: 1000.00 StockTo 1000.00
OrderAt: 900.00 EmerOrderAt 250.00

FROZEN
OnHand: 500.00 StockTo 500.00
OrderAt: 375.00 EmerOrderAt 125.00

Mail
OnHand: 200.00 StockTo 200.00
OrderAt: 190.00 EmerOrderAt 50.00

Water
OnHand: 300000.00 StockTo 300000.00
OrderAt: 225000.00 EmerOrderAt 75000.00
```

APPENDIX G

SWOS-129 PREPO FILES

```
NumberOfFiles: 9

============================================================================
============================================================================

Bonneyman

Transporter:  Comet
Latitude:  3   0 N
Longitude: 72 30 E

Cargo Destination:  7MEB
Cargo Routing:  2
PINTOINT
7MEB

Cargo:  2

M-1A1  58.000000
UNITEQ 7738.0

============================================================================
============================================================================

Altair

Transporter:  SL-7
Latitude:  36  50 N
Longitude: 76 20 W

Cargo Destination:  Wilmington
Cargo Routing:  1
Wilmington

Cargo:  1

UNITEQ 1.0

============================================================================
============================================================================

Algol

Transporter:  SL-7
Latitude:  29 30 N
Longitude: 95 45 W

Cargo Destination:  Corpus
Cargo Routing:  1
Corpus

Cargo:  1

UNITEQ 1.0

============================================================================
============================================================================

Antares
```

```
Transporter:  SL-7
Latitude:  33  0 N
Longitude: 81 35 W

Cargo Destination:  Wilmington
Cargo Routing:  1
Wilmington

Cargo:  1

UNITEQ 1.0

================================================================================
================================================================================

Bellatrix

Transporter:  SL-7
Latitude:  29 30 N
Longitude: 95 45 W

Cargo Destination:  Corpus
Cargo Routing:  1
Corpus

Cargo:  1

UNITEQ 1.0

================================================================================
================================================================================

Capella

Transporter:  SL-7
Latitude:  33  0 N
Longitude: 81 35 W

Cargo Destination:  Wilmington
Cargo Routing:  1
Wilmington

Cargo:  1

UNITEQ 1.0

================================================================================
================================================================================

Denebola

Transporter:  SL-7
Latitude:  40 40 N
Longitude: 74 10 W

Cargo Destination:  Wilmington
Cargo Routing:  1
Wilmington

Cargo:  1
```

```
UNITEQ 1.0

=========================================================================
=========================================================================

Pollux

Transporter:  SL-7
Latitude:  30  0 N
Longitude: 90  0 W

Cargo Destination:  Corpus
Cargo Routing:  1
Corpus

Cargo:  1

UNITEQ 1.0

=========================================================================
=========================================================================

Regulus

Transporter:  SL-7
Latitude:  30  0 N
Longitude: 90  0 W

Cargo Destination:  Corpus
Cargo Routing:  1
Corpus

Cargo:  1

UNITEQ 1.0
```

APPENDIX H

SWOS-129 SCENARIO FILES

```
SWOS129.scn
==========================================================================
Commods.mst
Trnsprts.mst
SWOS129.bse
SWOS129.unt
SWOS129.blk
SWOS129.rlk
SWOS129.tlk
SWOS129.ppo
==========================================================================

SWOS129.bse
==========================================================================
NumBases:  24

Guam
MASIRAH
DIEGO
PINTOINT
NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
Singapore
Rota
Sigonella
Okinawa
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus
Djibouti
Muscat
Bahrain
Dubai
==========================================================================

SWOS129.unt
==========================================================================
NumUnits:  13

MEF
CTF50
CTG51.1A
CTG51.1B
CTF51.2
AF-MASIRAH
AF-RIYADH
VP-1
VP-8
VQ-1
3ACR
82ABNDIV
7MEB
==========================================================================
SWOS129.blk
==========================================================================
NumUnits:  13

Unit:  MEF

Origin:  Okinawa

Unit:  CTF50

Origin:  NSCNorfolk

Unit:  CTG51.1A

Origin:  NSCNorfolk

Unit:  CTG51.1B

Origin:  NSCSanDiego

Unit:  CTF51.2

Origin:  NSCSanDiego

Unit:  AF-MASIRAH

Origin:  Guam

Unit:  AF-RIYADH

Origin:  PopeAFB

Unit:  VP-1

Origin:  DIEGO

Unit:  VP-8

Origin:  DIEGO

Unit:  VQ-1

Origin:  DIEGO

Unit:  3ACR

Origin:  FtBliss

Unit:  82ABNDIV

Origin:  FtBragg

Unit:  7MEB

Origin:  Guam

==========================================================================

SWOS129.rlk
```

```
==========================================================================
NumBases:  37

Base:  Guam
NumNodes:  0


Base:  MASIRAH
NumNodes:  0


Base:  DIEGO
NumNodes:  0


Base:  PINTOINT
NumNodes:  0


Base:  NWSEarle
NumNodes:  11

MOTBayonne
NSCNorfolk
MOTOakland
NWSConcord
NSCSanDiego
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  NWSConcord
NumNodes:  11

MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
NWSEarle
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  MOTBayonne
NumNodes:  11

NSCNorfolk
MOTOakland
NSCSanDiego
NWSEarle
NWSConcord
FtBragg
PopeAFB
Wilmington
```

```
FtBliss
HollomanAFB
Corpus

Base:  NSCNorfolk
NumNodes:  11

MOTOakland
NSCSanDiego
NWSEarle
NWSConcord
MOTBayonne
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  MOTOakland
NumNodes:  11

NSCSanDiego
NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  NSCSanDiego
NumNodes:  11

NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  Singapore
NumNodes:  0


Base:  Rota
NumNodes:  0


Base:  Sigonella
NumNodes:  0
```

```
Base:  Okinawa
NumNodes:  0


Base:  FtBragg
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  PopeAFB
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  Wilmington
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
FtBliss
HollomanAFB
Corpus

Base:  FtBliss
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
```

```
PopeAFB
Wilmington
HollomanAFB
Corpus

Base:  HollomanAFB
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
Wilmington
FtBliss
Corpus

Base:  Corpus
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB

Base:  Djibouti
NumNodes:  0


Base:  Muscat
NumNodes:  0


Base:  Bahrain
NumNodes:  0


Base:  Dubai
NumNodes:  0


Base:  MEF
NumNodes:  0


Base:  CTF50
NumNodes:  0


Base:  CTG51.1A
```

```
NumNodes:  0


Base:  CTG51.1B
NumNodes:  0


Base:  CTF51.2
NumNodes:  0


Base:  AF-MASIRAH
NumNodes:  0


Base:  AF-RIYADH
NumNodes:  0


Base:  VP-1
NumNodes:  0


Base:  VP-8
NumNodes:  0


Base:  VQ-1
NumNodes:  0


Base:  3ACR
NumNodes:  0


Base:  82ABNDIV
NumNodes:  0


Base:  7MEB
NumNodes:  0
================================================================================

SWOS129.tlk
================================================================================
NumBases:  37

Base:  Guam
NumNodes:  0


Base:  MASIRAH
NumNodes:  3

VP-8
VQ-1
AF-MASIRAH

Base:  DIEGO
NumNodes:  1
```

```
VP-1

Base:  PINTOINT
NumNodes:  3

7MEB
3ACR
82ABNDIV

Base:  NWSEarle
NumNodes:  11

MOTBayonne
NSCNorfolk
MOTOakland
NWSConcord
NSCSanDiego
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  NWSConcord
NumNodes:  11

MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
NWSEarle
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  MOTBayonne
NumNodes:  11

NSCNorfolk
MOTOakland
NSCSanDiego
NWSEarle
NWSConcord
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  NSCNorfolk
NumNodes:  11

MOTOakland
NSCSanDiego
```

```
NWSEarle
NWSConcord
MOTBayonne
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  MOTOakland
NumNodes:  11

NSCSanDiego
NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  NSCSanDiego
NumNodes:  10

NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NWSEarle
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB

Base:  Singapore
NumNodes:  0


Base:  Rota
NumNodes:  0


Base:  Sigonella
NumNodes:  0


Base:  Okinawa
NumNodes:  0


Base:  FtBragg
NumNodes:  11

NWSEarle
NWSConcord
```

```
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
PopeAFB
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  PopeAFB
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
Wilmington
FtBliss
HollomanAFB
Corpus

Base:  Wilmington
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
FtBliss
HollomanAFB
Corpus

Base:  FtBliss
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
Wilmington
HollomanAFB
Corpus

Base:  HollomanAFB
NumNodes:  11

NWSEarle
NWSConcord
```

```
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
Wilmington
FtBliss
Corpus

Base:  Corpus
NumNodes:  11

NWSEarle
NWSConcord
MOTBayonne
NSCNorfolk
MOTOakland
NSCSanDiego
FtBragg
PopeAFB
Wilmington
FtBliss
HollomanAFB

Base:  Djibouti
NumNodes:  0


Base:  Muscat
NumNodes:  0


Base:  Bahrain
NumNodes:  0


Base:  Dubai
NumNodes:  0


Base:  MEF
NumNodes:  0


Base:  CTF50
NumNodes:  0


Base:  CTG51.1A
NumNodes:  0


Base:  CTG51.1B
NumNodes:  0


Base:  CTF51.2
NumNodes:  0
```

```
Base:  AF-MASIRAH
NumNodes:  1

MASIRAH

Base:  AF-RIYADH
NumNodes:  0


Base:  VP-1
NumNodes:  1

DIEGO

Base:  VP-8
NumNodes:  1

MASIRAH

Base:  VQ-1
NumNodes:  1

MASIRAH

Base:  3ACR
NumNodes:  1

PINTOINT

Base:  82ABNDIV
NumNodes:  1

PINTOINT

Base:  7MEB
NumNodes:  1

PINTOINT
==========================================================================

SWOS129.ppo
==========================================================================
NumPrepo:  9

Bonneyman
Altair
Algol
Antares
Bellatrix
Capella
Denebola
Pollux
Regulus
```

APPENDIX I

SURGE AND SUSTAINMENT SIMULATION USERS GUIDE

TABLE OF CONTENTS

I. STARTING S3

Welcome to the modeling of logistics with the Surge and Sustainment Simulation, S3, Version 1.0! With this program you should be able to simulate a theater logistics system for use with a wargame that does not normally support logistics. When executed, S3 creates a network of *Bases* and *Units* which consume, order and provide *Commodities*. These *Commodities* represent the means making war, the men and material that make military campaigns possible. These Commodities are moved from Base to Base and Unit by *Transporters* that model the various cargo vehicles typical of the major modes of transportation (air, sea, road and rail).

The user, normally a game administrator, such as an umpire or referee, will execute this simulation in coordination with a separate wargame. Providing the link between the game and the logistics simulation, the user ensures that game events are reflected in model inputs and model output is reflected in players decisions and game outcomes. In a sense, S3 is a logistics calculator that transforms the game inputs into logistical results that, in turn, should influence game events.

Before execution is possible, the various Base, Units, Commodities and Transporters must be created and brought together in a unifying *Scenario*. S3 features a built-in

Scenario Editor which enables the creation of these program entities. Base, Units, Commodities, Transporters, and several other constructs may be built with the Scenario Editor and used in countless Scenarios. Each entity, such as a Transporter or Commodity represents a generic model of its actual counterpart. When a Scenario is created it is populated with multiple instances of these generic objects. Thus, Bases, Transporters, and the like may be reused again and again in new and different configurations allowing for flexibility in Scenario generation.

A. SYSTEM REQUIREMENTS

Version 1.0 runs on a Unix-based system such as a Sun Workstation. You will need to ensure that MODSIM II version 1.9.1 is correctly loaded and accessible from the directory in which you intend to execute S3. The workstation does not require OpenWindows, XWindows or any other windows system for Unix machines in order to run the program. If you wish to execute S3 while in a windows environment, you should open a Command Tool or similar window device.

NOTE

Since Unix and Modsim II are case sensitive, so is S3. You must be careful to use the correct cases when typing at system and program prompts.

To start the Surge and Sustainment Simulation type "S3" at the main prompt or at the prompt of a Command Tool and hit **<ENTER>**. You will enter the **Main Menu** (Figure 1).

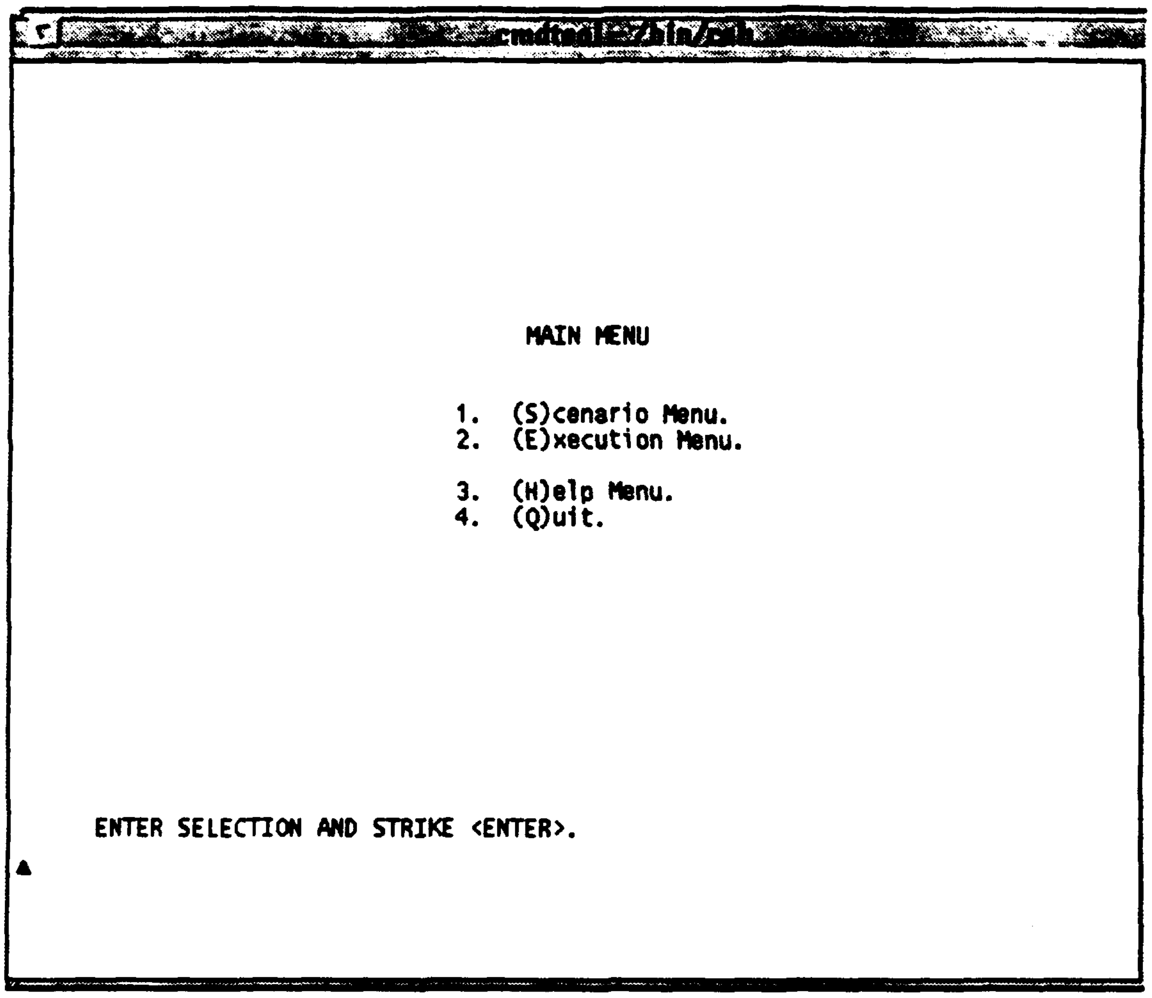

```
MAIN MENU

1.  (S)cenario Menu.
2.  (E)xecution Menu.

3.  (H)elp Menu.
4.  (Q)uit.



ENTER SELECTION AND STRIKE <ENTER>.
```

Figure 1 The Main Menu

B. THE MENU SYSTEM

All S3 menus share a common structure. In all cases, the commands that may be executed from a menu will displayed. In order to execute a particular command, press the key indicated by the parentheses, "()". The key press will work in either upper or lower case. In some cases, commands will appear in enumerated lists. These commands may be executed by pressing the corresponding number key or the key indicated by the parentheses.

S3 is essentially divided into two main sections, the Execution Menu and the Scenario Editor. The Main Menu is the dividing line between these two sections. From the Main Menu you can select the Execution Menu or the Scenario Menu. At the Main Menu you will be presented with an enumerated list of commands. Pressing "1" or "S" will call the Scenario Menu. Pressing "2" or "E" will call the Execution Menu. Pressing "3" or "Q" will quit the program. Pressing "4" or "H" will call an appropriate help screen (Help is not available in Version 1.0).

II. RUNNING A SCENARIO

A. THE EXECUTION MENU

To run a scenario you must select the Execution Menu (Figure 2) at the **Main Menu** prompt by pressing "**2**" OR "**E**". At the Scenario Menu, you may execute several functions. You may select a Scenario, execute a selected Scenario, or reset the Scenario Clock.

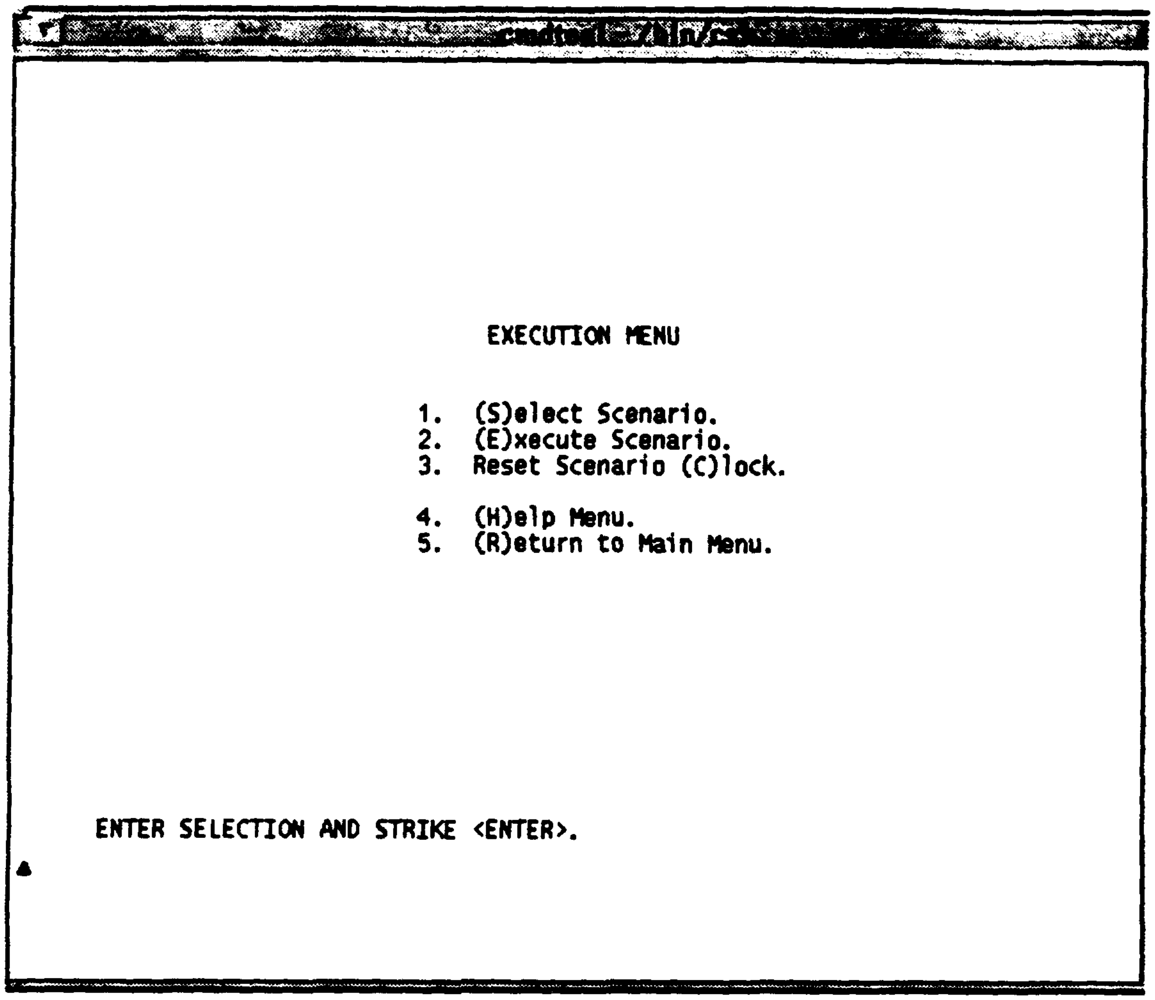
1. Selecting a Scenario

When you enter the Scenario Menu, the first Scenario on the Scenario List is automatically selected for execution. This is to prevent a runtime error from occurring by attempting execute when no Scenario has been selected. To choose a Scenario:

1. From the **Execution Menu**, press "1" or "S."
2. S3 will respond by displaying a list of all the Scenarios that have already been built. If none are listed, you must construct a Scenario for S3 to run before proceeding.
3. Type the desired Scenario Name, exactly as it appears in the listing, and press **<ENTER>**.
4. The Scenario has now been selected, you will be returned to the Execution Menu.

2. Executing a Scenario

Once you have selected a Scenario, you may begin execution. To load the Scenario, press "**2**" or "**E**" from the **Execution Menu**. After a momentary pause while the Scenario is being loaded, you will enter the **Time Step Menu**.



```
                         EXECUTION MENU

                    1.  (S)elect Scenario.
                    2.  (E)xecute Scenario.
                    3.  Reset Scenario (C)lock.

                    4.  (H)elp Menu.
                    5.  (R)eturn to Main Menu.




  ENTER SELECTION AND STRIKE <ENTER>.
```

Figure 2 The Execution Menu

3. Resetting the Scenario Clock

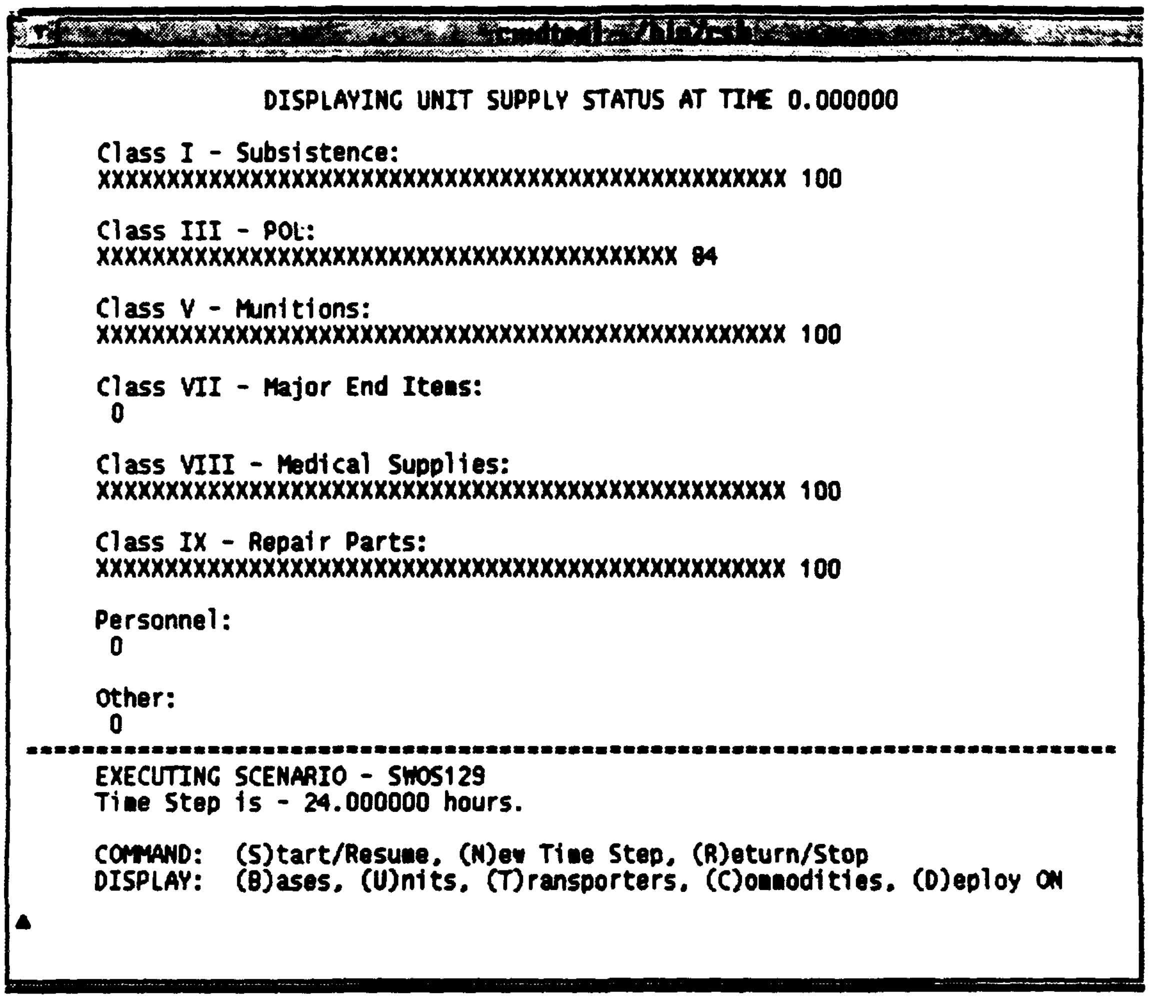
Currently, the simulation clock is not reset when you quit a Scenario. If you intend to execute another Scenario or re-execute the one you have just left, you must reset the simulation clock. From the **Execution Menu**, press "3" or "C."

B. THE TIME STEP MENU

The Time Step Menu (Figure 3) is where execution of S3 occurs. At this menu a graphic representation of Unit Supply Status is displayed. From the Time Step Menu you may start or resume scenario execution, change the Time Step, quit the execution, or display Bases, Units, Transporter, or Commodities. You may also choose to view the Unit Supply Status Display or the Unit Deployment Status Display.

1. Changing the Time Step

The *Time Step* is an important concept in simulating logistics with S3. Essentially, the Time Step is the amount of simulation time that S3 will run before allowing you to view the progress of the logistics system or to make inputs. When the you start or continue a scenario execution, S3 will run on "autopilot" using the current inputs for the duration of a Time Step. For example, if you are certain that no inputs or displays will be required for the next three simulated days, you could set the Time Step to 72.0. Every time the you type "S" to Start/Resume the simulation, S3 will now run for 72 simulated hours before allowing you to enter a command. This is S3's way of jumping ahead in time. At the



```
                    DISPLAYING UNIT SUPPLY STATUS AT TIME 0.000000

     Class I - Subsistence:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Class III - POL:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 84

     Class V - Munitions:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Class VII - Major End Items:
      0

     Class VIII - Medical Supplies:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Class IX - Repair Parts:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Personnel:
      0

     Other:
      0
 ================================================================================
     EXECUTING SCENARIO - SWOS129
     Time Step is - 24.000000 hours.

     COMMAND:  (S)tart/Resume, (N)ew Time Step, (R)eturn/Stop
     DISPLAY:  (B)ases, (U)nits, (T)ransporters, (C)ommodities, (D)eploy ON
▲
```

Figure 3 The Time Step Menu

outset of any scenario, the Time Step is automatically set to a default value of 24.0 simulated hours. To change Time Step:

1. From the **Time Step Menu**, press "**N**".

2. Type in the desired Time Step when prompted. The format is **X.X** (eg., 168.0). Press **<ENTER>**.

3. The new Time Step has now been set. The new Time Step should appear above the COMMAND line on the Time Step Menu.

2. Displaying Bases or Units

One of the most important features of S3 is the ability to view the supply status of a Base or Unit at any Time Step. During execution, this is accomplished by displaying the Base or Units from the Time Step Menu. The display will provide all of the vital information concerning a Base or Unit including Name, Position, Groups, Deployment Status, Combat Intensity, Port Information, and Inventory. You can:

1. Display a single Base or Unit.
2. Display all Bases or Units.
3. Modify a Base or Unit.

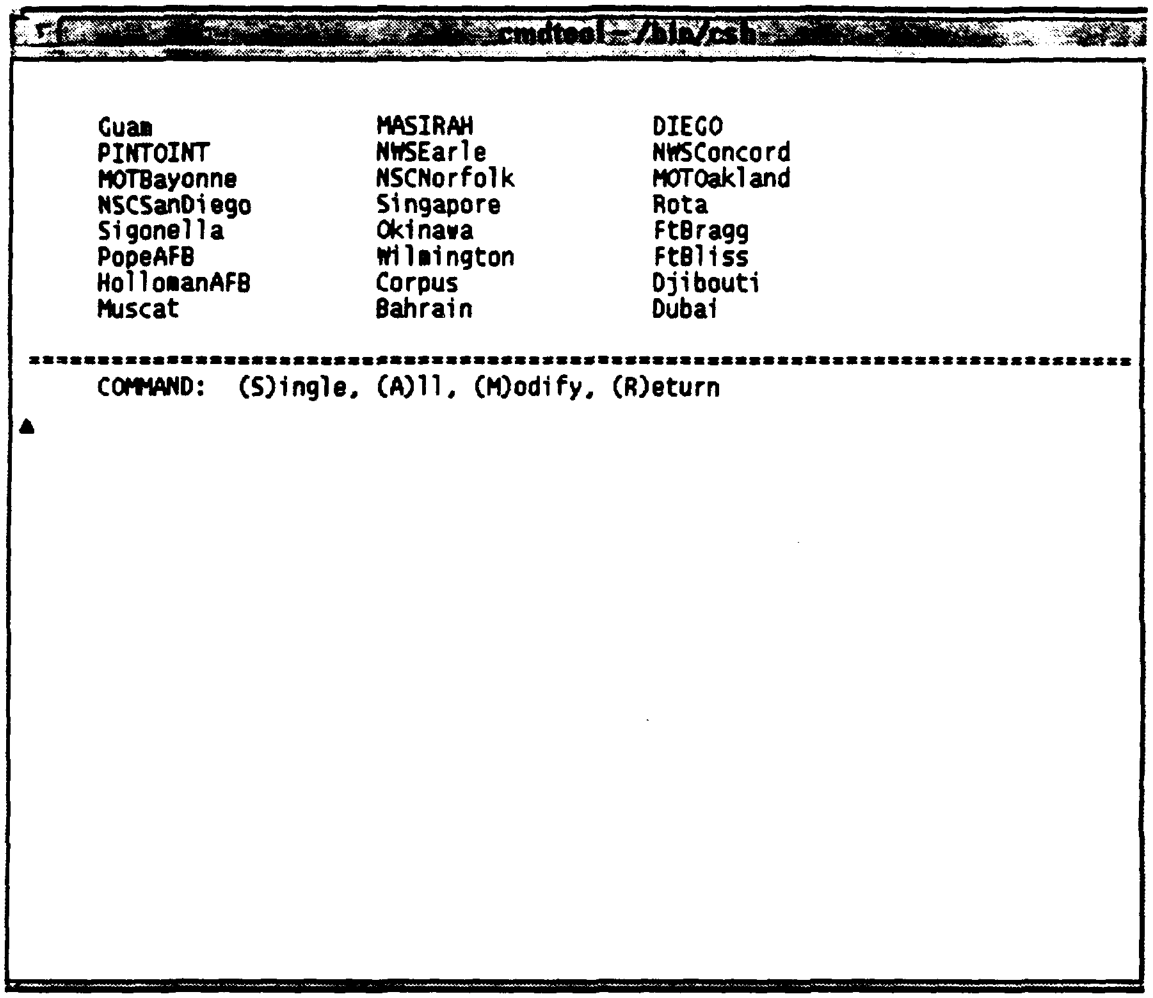
a. *Viewing A Single Base or Unit*

In order to view a particular base in a scenario:

1. From the **Time Step Menu** press "**B**" to display Bases or "**U**" to display Units.

2. A list of all the Bases or Units in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 4. To view a single Base or Unit, press "**S**".

3. S3 will then ask for the Name of the Base or Unit that you wish to display. Type the Name in exactly as it appears in the list above. Pay special attention to upper and lower case characters as S3 is case sensitive. When the Name of the Base or Unit has been



```
        Guam                MASIRAH             DIEGO
        PINTOINT            NWSEarle            NWSConcord
        MOTBayonne          NSCNorfolk          MOTOakland
        NSCSanDiego         Singapore           Rota
        Sigonella           Okinawa             FtBragg
        PopeAFB             Wilmington          FtBliss
        HollomanAFB         Corpus              Djibouti
        Muscat              Bahrain             Dubai

==============================================================================
        COMMAND:  (S)ingle, (A)ll, (M)odify, (R)eturn
▲
```

Figure 4 Base Listing

typed, Press **<ENTER>**.

4. A Display of the Base or Units similar to Figure 5 or 6 will appear on the screen. If you wish to return to the previous Base listing, press "**Y**" when prompted.

b. Viewing All Bases or Units In The Scenario

Rather than repetitively entering the names of the Bases or Units in a scenario when you wish to view them, you can view all of the Bases or Units sequentially. The display will be identical to the display of a single Base or Unit, however, all of the Bases will be displayed in turn. To display all Bases or Units:

1. From the **Time Step Menu** press "**B**" to display Bases or "**U**" to display Units.

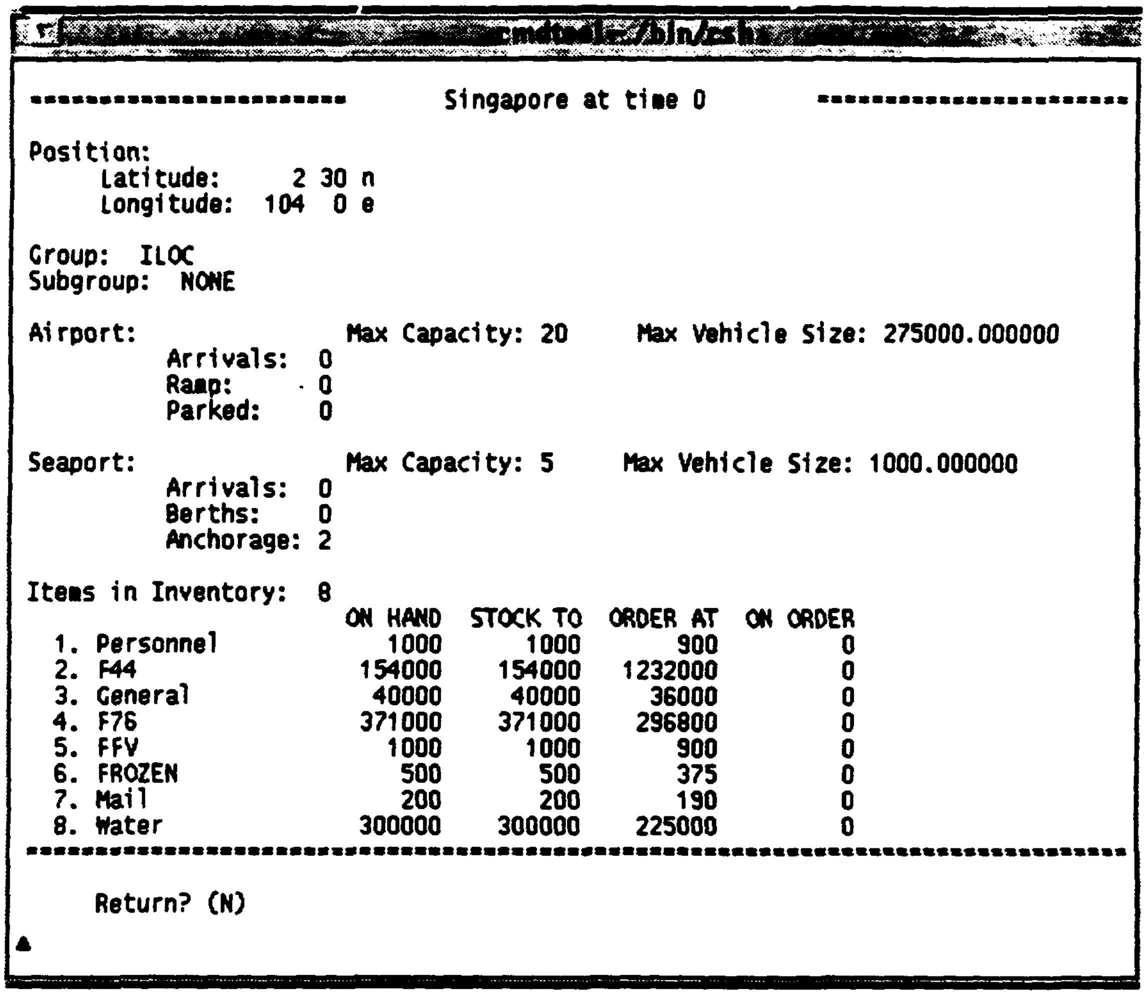
2. A list of all the Bases or Units in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 4. To view a all Bases or Units, press "**A**".

3. A display of the first Base or Units on the listing will appear on the screen. When you wish to view the next Base or Unit on the list, press **any key** to continue. When you wish to return to the previous Base listing, press "**Y**" when asked to if you wish to return.

c. Modifying A Base

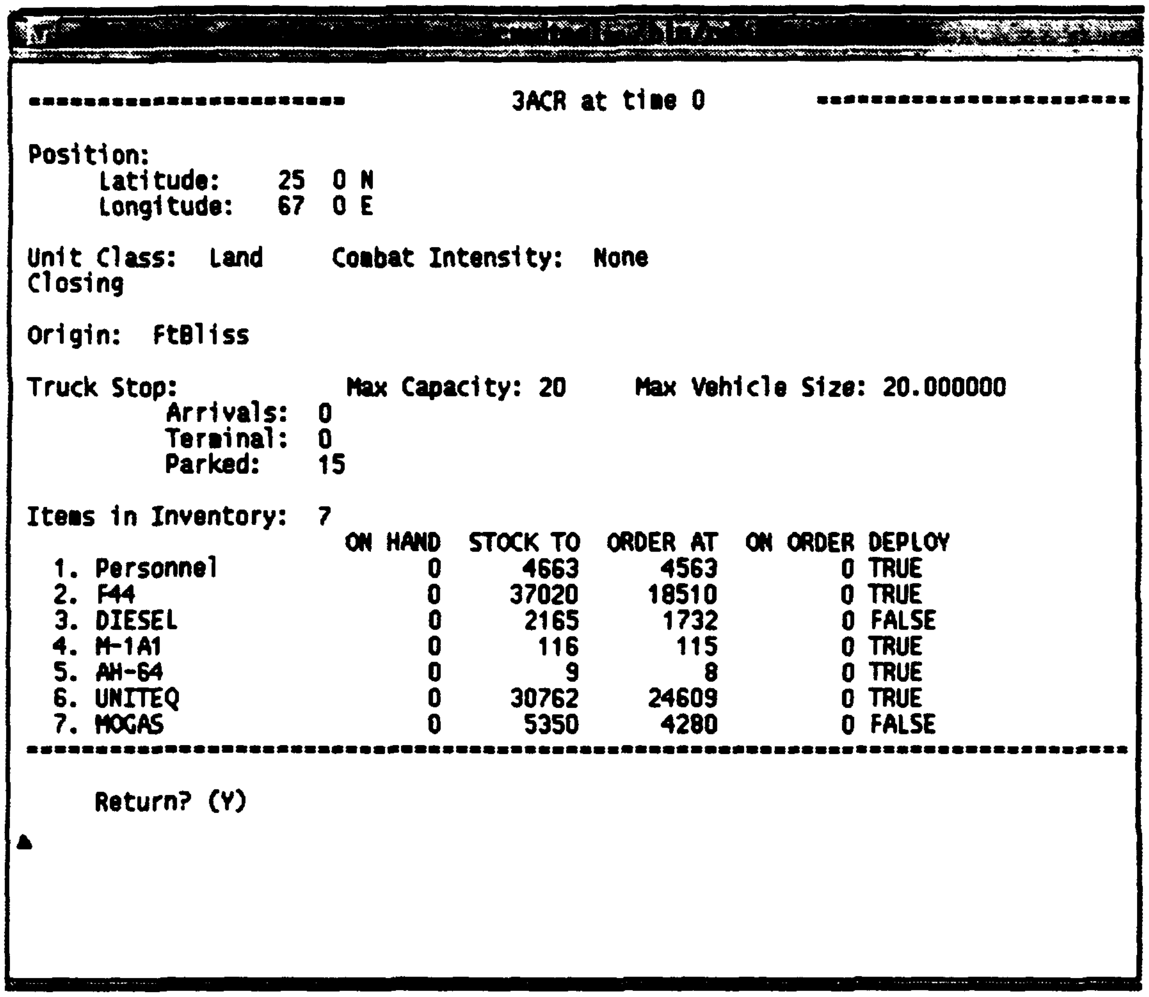
During the execution of a scenario, you may modify the configuration of a Base. The following lists the modifications that can be made to Base during scenario execution:

1. Toggle Port ON or OFF
2. Change Port Capacity
3. Change Port Maximum Vehicle size
4. Add or Delete a Base from Transportation Networks
5. Add or Delete a Transporter
6. Add or Edit a Commodity in Inventory
7. Change Position



```
======================        Singapore at time 0       ======================

Position:
     Latitude:     2 30 n
     Longitude:  104  0 e

Group:  ILOC
Subgroup:  NONE

Airport:                    Max Capacity: 20     Max Vehicle Size: 275000.000000
          Arrivals:  0
          Ramp:      0
          Parked:    0

Seaport:                    Max Capacity: 5     Max Vehicle Size: 1000.000000
          Arrivals:  0
          Berths:    0
          Anchorage: 2

Items in Inventory:  8
                            ON HAND  STOCK TO  ORDER AT  ON ORDER
  1. Personnel                 1000      1000       900         0
  2. F44                     154000    154000   1232000         0
  3. General                  40000     40000     36000         0
  4. F76                     371000    371000    296800         0
  5. FFV                       1000      1000       900         0
  6. FROZEN                     500       500       375         0
  7. Mail                       200       200       190         0
  8. Water                   300000    300000    225000         0
==============================================================================

     Return? (N)
```

Figure 5 The Base Display



```
=======================            3ACR at time 0         ======================

Position:
     Latitude:     25  0 N
     Longitude:    67  0 E

Unit Class:  Land     Combat Intensity:  None
Closing

Origin:  FtBliss

Truck Stop:            Max Capacity: 20     Max Vehicle Size: 20.000000
          Arrivals:  0
          Terminal:  0
          Parked:    15

Items in Inventory:  7
                       ON HAND  STOCK TO  ORDER AT  ON ORDER DEPLOY
  1. Personnel               0      4663      4563         0 TRUE
  2. F44                     0     37020     18510         0 TRUE
  3. DIESEL                  0      2165      1732         0 FALSE
  4. M-1A1                   0       116       115         0 TRUE
  5. AH-64                   0         9         8         0 TRUE
  6. UNITEQ                  0     30762     24609         0 TRUE
  7. MOGAS                   0      5350      4280         0 FALSE
===============================================================================

     Return? (Y)
```

Figure 6 The Unit Display

It is important to note that modifications to a Base or Unit are NOT permanent. These changes will only remain in effect during the current execution. If permanent changes to a Base or Unit are desired, edit the Base or Unit from the Scenario Editor.

To enter the **Modification Display of a Base**:

1. From the **Time Step Menu** press "**B**" to display Bases.

2. A list of all the Bases in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 4. To modify a Base press "**M**".

3. S3 will then ask for the Name of the Base that you wish to display. Type the Name in exactly as it appears in the list above. Pay special attention to upper and lower case characters as S3 is case sensitive. When the Name of the Base has been typed, Press **<ENTER>**.

4. A display of the Base will appear on the screen with a list of commands. To perform a modification, press one of the indicated keys. When you wish to return to the previous Base listing, press "**R**" at the command prompt.

(1) Toggling a Port ON or OFF. Toggling a Port OFF effectively removes it from the consideration of the Logistics Manager. When the Logistics Manager builds a route for a Shipment, it determines if Bases can communicate with like modes of transportation. If a Base Port is OFF, the Logistics Manager will not consider the Port when determining if a Base can communicate with another Base. Thus, the routing of Shipment is directly affected by the existence of Ports.

Nevertheless, all Shipments or Transporters already enroute to a Port will arrive and unload normally. Any Shipments already at the Port's Loading Docks will continue to their destinations. No future Shipments will be sent by way of a Port that has been turned OFF.

Of course, the opposite is true of Ports that are turned ON. Special care should be taken to ensure that the Port has a satisfactory Maximum Capacity and Vehicle size when turning a Port On. If the Capacity of a newly constructed Port is set still to its default value of zero when a Transporter arrives, it will remain in the Arrival Queue and never enter the Berth Queue.

To toggle a Port ON or OFF:

1. From the **Base Modification Display** press **"P"** to modify the Port configuration of the Base.

2. A list of the Ports currently in existence at the Base will be dispayed. Additionally, a set of choices will be listed below. Select the Port you wish to modify by pressing the indicated key.

3. Another list of choices will be displayed. To toggle the selected Port ON or OFF press **"E"**.

4. The list of Ports will be again be displayed with the list of Port choices. If you wish to return to the **Base Modification Display**, press **"R"**.

(2) Changing a Port's Capacity. The capacity of a Port is the number of Transporters that it may load or unload simultaneously. For example, the number of berths that may be active at a seaport is an effective estimator of a Port capacity for S3. An active Port must have a Capacity that is greater than zero for the Port to function properly. To

change a Port Capacity:

1. From the **Base Modification Display** press "**P**" to modi fy the Port configuration of the Base.

2. A list of the Ports currently in existence at the Base will be dislayed. Additionally, a set of choices will be listed below. Select the Port you wish to modify by pressing the indicated key.

3. Another list of choices will be displayed. To change the selected Port's Capacity, press "**C**".

4. Type in the new Capacity of the Port at the prompt. The format of the entry is **XX** (eg., 2 or 34). When the desired Capacity has been typed, press **<ENTER>**.

5. The list of Ports will be again be displayed with the list of Port choices. If you wish to return to the **Base Modification Display**, press "**R**".

(3) Changing a Port's Maximum Vehicle Size. The Maximum Vehicle Size of a Port is the size of the largest Transporter that the Port can accommodate. For a seaport, this would be length of the largest vessel that may be unloaded at the Port. This does not necessarily mean that Maximum Vehicle Capacity is the length of the largest berth since a ship might unload to lighters in the middle of a harbor.

In Version 1.0, the Maximum Vehicle Size constraint is not in effect. To change a Port Maximum Vehicle Size:

1. From the **Base Modification Display** press "**P**" to modify the Port configuration of the Base.

2. A list of the Ports currently in existence at the Base will be displayed. Additionally, a set of choices will be listed below. Select the Port you wish to modify by pressing the indicated key.

3. Another list of choices will be displayed. To change the selected Port's Maximum Vehicle Size, press "**S**".

4. Type in the new Maximum Vehicle Size of the Port at the prompt. The format of the entry is **X.X** (eg., 1200.0). When the desired Size has been typed, press **<ENTER>**.

5. The list of Ports will be again be displayed with the list of Port choices. If you wish to return to the **Base Modification Display**, press "**R**".

(4) Adding or Deleting a Base from a Network. Networks only apply to Rail Yards and Truck Stops since trucks and trains are generally restricted to the confines of a single land mass. During execution, you can add or delete a Base or Unit from a particular Port Network. This feature is included primarily to allow the construction of Port facilities that did not exist prior to execution of the scenario. It is far easier to construct a Port transportation network during the construction of a Scenario. To add or delete a Base from a Network:

1. From the **Base Modification Display** press "**P**" to modify the Port configuration of the Base.

2. A list of the Ports currently in existence at the Base will be displayed. Additionally, a set of choices will be listed below. Select the Port you wish to modify by pressing the indicated key.

3. Another list of choices will be displayed. To change the selected Port Network, press "**N**".

4. The list of all the Bases currently in the Network will be displayed with a set of choices. To add a Base to this list, press "**A**". To delete a Base, press "**S**". If you wish to return, press "**R**".

5. If you selected "Add", you will be asked to provide Name of a Base or Unit. Type the Base or Unit Name, paying special attention to character case and press **<ENTER>**.

6. If you selected "Subtract", you will be asked to provide a Base or Unit Name from the previously displayed list. Type the Base or Unit Name at the prompt and press **<ENTER>**

(5) Adding or Deleting a Transporter. Transporters may be added or deleted in order to compensate for an oversight in Scenario construction or an unexpected decision by players. To add or delete a Transporter:

1. From the **Base Modification Display** press "**P**" to modify the Port configuration of the Base.

2. A list of the Ports currently in existence at the Base will be displayed. Additionally, a set of choices will be listed below. Select the Port you wish to modify by pressing the indicated key.

3. Another list of choices will be displayed. To add or delete a Transporter, press "**T**".

4. Another list of choices will be displayed. To add a Transporter, press "**A**". To delete a Transporter, press "**D**".

5. If you selected "Add":

   a. A list of all the Transporters types in the Scenario will be displayed. You will be asked to provide a Transporter Name. Type the Transporter Name, paying special attention to character case and press **<ENTER>**.

   b. You will then be asked the number of Transporters of the selected type to add to the Port. Type the number of Transporters and press **<ENTER>**. The format is **XX** (eg., 2 or 34).

7. If you selected "Subtract":

   a. A list of Transporters at the Port will be displayed. You will be asked to provide the Name of a Transporter that you wish to delete. Type the Transporter Name, paying special attention to character case and press **<ENTER>**.

   b. You will then be asked to input the Vehicle ID of the Transporter that you wish to delete. Type the Vehicle ID of the Transporter and press

<ENTER>. The format is XX (eg., 2 or 34).

8. You will be returned to **Base Modification Display**.

(6) Adding or editing a Commodity from Inventory. You may also add or edit a Commodity from a Base Inventory to compensate for oversights in the Scenario construction process. To add or edit a Commodity:

1. From the **Base Modification Display** press "I" to modify the Inventory of the Base.

2. A list of choices will be displayed. To add a Commodity, press "**A**". To edit a Commodity already in Inventory, press "**E**".

3. If you selected "Add Commodity":

   a. A list of all the Commodities in the scenario will the displayed. You will be asked to input the Name of a Commodity that you wish to add to the Base Inventory. Type in the Name of the desired Commodity and press **<ENTER>**.

   b. You will then be asked a series of questions concerning the new Commodity. At each prompt, type the appropriate real number (**X.X**) and press **<ENTER>**.

   c. The Commodity will then be added to the Base Inventory. You will be asked if you wish to add another Commodity. To return, press "**N**". To add another Commodity, **press any other key**.

4. If you selected "Edit Commodity":

   a. You will be asked to input the Name of the Commodity to edit. Type in the Name of the desired Commodity and press **<ENTER>**.

   b. A list of choices will be displayed. To change the amount On Hand, press "O". To change the Commodity Stocking Objective, press "S". To change the Order Point, press "O". To return, press "**R**".

c. You will then be asked to enter the selected value. At the prompt, type the appropriate real number (**X.X**) and press **<ENTER>**. You will be returned to the previous list of commands.

5. You will be automatically returned to the **Base Display Menu** when the procedure is completed.

(7) Changing a Base's Position. During a scenario, it is unlikely that a Base will ever change position. However, this feature has been provided for those who may wish to model mobile base (such as a CLF "gasoline alley") or have made errors during Scenario construction. To Modify a Base's Position:

1. From the **Base Modification Display** press "**S**" to modify the Status of the Base.
2. A list of choices will be displayed. To change a Base's Position, press "**P**". Proceed as described in Section III.E.1 **Building a Base**.
3. You will be returned to the previous list of choices. To return to the **Base Modification Display**, press "**R**".

d. Modifying a Unit

During the execution of a scenario, you may also modify the configuration of a Unit. This modification procedure is almost identical to the Base modification procedure. The following additional modifications can be made to a Unit during Scenario execution:

1. Change a Unit Combat Intensity
2. Change a Unit Closure Status
3. Activate a Unit

It is important to note that modifications to a Unit are NOT permanent. These changes will only remain in effect during the current execution. If permanent changes to a Base or Unit are desired, edit the Base or Unit from the Scenario Editor.

To enter the **Modification Display of a Unit**:

1. From the **Time Step Menu** press "**U**" to display Bases.

2. A list of all the Units in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 4. To modify a Unit press "**M**".

3. S3 will then ask for the Name of the Unit that you wish to display. Type the Name, exactly as it appears in the list above. Pay special attention to the upper and lower case characters as S3 is case sensitive. When the Name of the Unit has been typed, Press **<ENTER>**.

4. A display of the Unit, similar to Figure 6, will appear on the screen with a list of commands. To perform a modification, press one of the indicated keys. When you wish to return to the previous Unit listing, press "**R**" at the command prompt.

(1) Changing a Unit Combat Intensity. Changing a Unit's Combat Intensity is one of the more routine modifications that will be made during an S3 execution. You will wish to change Combat Intensity for active Units whenever there is a game driven change in the tempo of operations. Typical examples are the initiation of air strikes, the start of a ground offensive, the declaration of hostilities, etc....

To change a Unit's Combat Intensity:

1. From the **Unit Modification Display** press "**S**" to modify the Status of the Unit.

2. A list of choices will be displayed. To change Combat Intensity, press "**I**". To return, press "**R**".

3. Another list of choices will be displayed. Select the desired Combat Intensity by pressing the indicated key.

4. You will be automatically returned to the previous list of choices.

(2) Changing a Unit Closure Status. In certain instances, it may be necessary to change a Unit Closure Status. This is likely to occur if a Unit is forced to consume Commodities before it reaches its closure criterion. For example, a ground Unit is sent into combat when half of its equipment arrives in theater despite the fact that the Unit is not considered fully deployed. The only way to force the Unit to begin consumption is to change its Closure Status to "In Place" vice "Closing". This would enable the consumption of Commodities and change the routing of all subsequent Shipments. All deployment Shipments already in transit will continue on their previously designated routes. Of course, should it become necessary, the process may be reversed and an Unit which has deployed can be changed to "Closing".

To change a Unit's Closure Status:

1. From the **Unit Modification Display** press "**S**" to modify the Status of the Unit.
2. A list of choices will be displayed. To change Closure Status, press "**S**". To return, press "**R**".
3. Another list of choices will be displayed. Select the desired Closure Status by pressing the indicated key.
4. You will be automatically returned to the previous list of choices.

(3) Activating a Unit. Another modification which will occur during a typical execution is the activation of a Unit. A deploying Unit typically starts a scenario in an inactive state so that the decision to mobilize may be

coordinated with game events and or the deployment of other Units. Units that are not deployed and do not start the game in an active status may be activated at any time. To activate a Unit:

1. From the **Unit Modification Display** press "S" to modify the Status of the Unit.

2. A list of choices will be displayed. To activate the Unit, press "**A**". To return, press "**R**".

3. The Unit is then activated. You will be automatically returned to the previous list of choices.

NOTE
You may only activate a Unit once. Activating a Unit that is already active may result in a runtime error or other program malfunction.

3. Displaying Transporters

Another helpful feature of S3 is the ability to display individual Transporter information. Information such as Position, Status, and Cargo may be displayed on the screen. During an S3 execution, the following actions may be performed on Transporters:

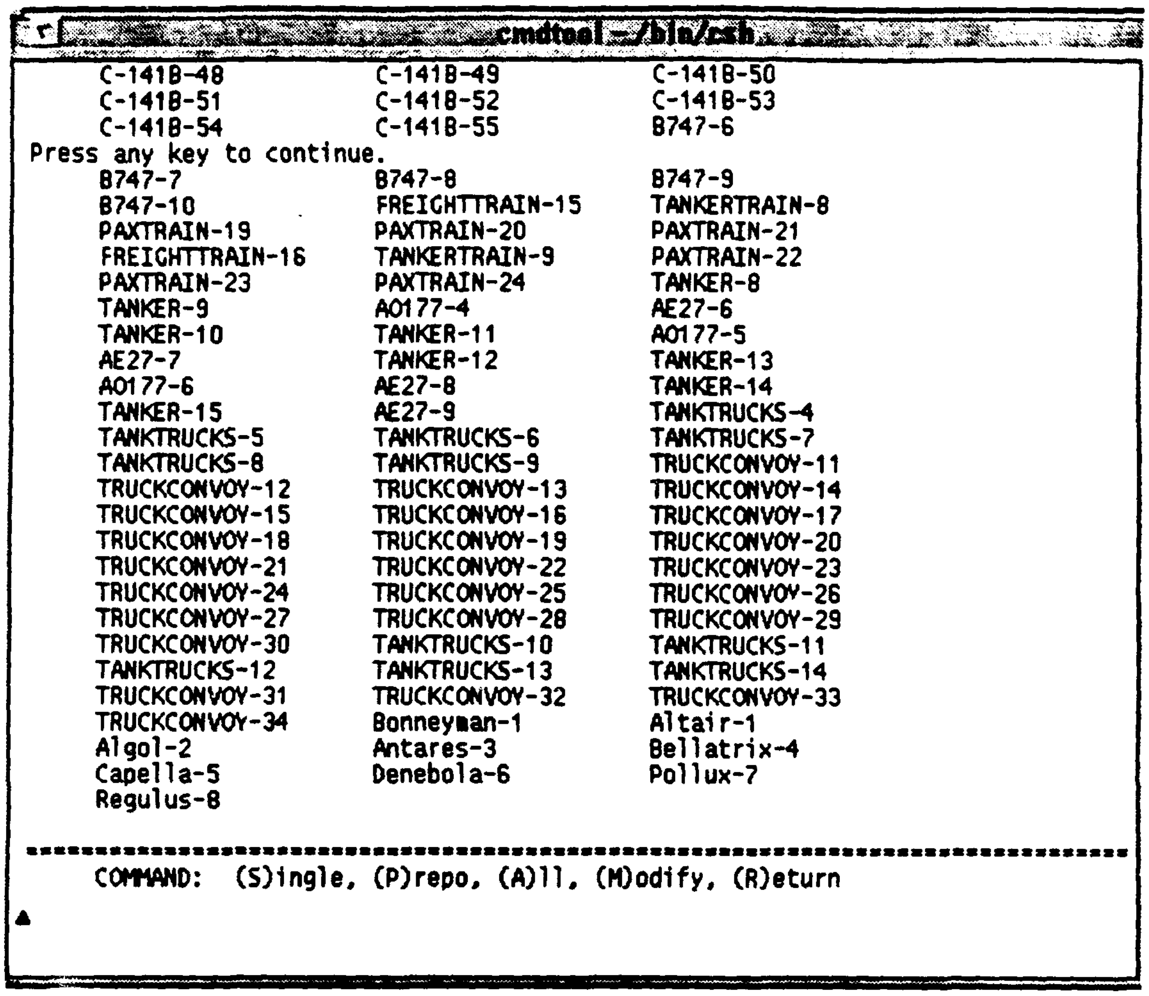
1. View a Single Transporter
2. View All Transporters in a Scenario
3. Activate a Prepositioned Transporter
4. Destroy a Transporter

a. Viewing A Single Transporter

To view a particular Transporter:

1. From the **Time Step Menu** press "**T**".

2. A list of all the Transporters in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 7. Each Transporter is identified by Name and Vehicle ID. The Vehicle ID is a single integer that comes after the Transporter



cmdtool -/bin/csh

```
          C-141B-48           C-141B-49           C-141B-50
          C-141B-51           C-141B-52           C-141B-53
          C-141B-54           C-141B-55           B747-6
Press any key to continue.
          B747-7              B747-8              B747-9
          B747-10             FREIGHTTRAIN-15     TANKERTRAIN-8
          PAXTRAIN-19         PAXTRAIN-20         PAXTRAIN-21
          FREIGHTTRAIN-16     TANKERTRAIN-9       PAXTRAIN-22
          PAXTRAIN-23         PAXTRAIN-24         TANKER-8
          TANKER-9            A0177-4             AE27-6
          TANKER-10           TANKER-11           A0177-5
          AE27-7              TANKER-12           TANKER-13
          A0177-6             AE27-8              TANKER-14
          TANKER-15           AE27-9              TANKTRUCKS-4
          TANKTRUCKS-5        TANKTRUCKS-6        TANKTRUCKS-7
          TANKTRUCKS-8        TANKTRUCKS-9        TRUCKCONVOY-11
          TRUCKCONVOY-12      TRUCKCONVOY-13      TRUCKCONVOY-14
          TRUCKCONVOY-15      TRUCKCONVOY-16      TRUCKCONVOY-17
          TRUCKCONVOY-18      TRUCKCONVOY-19      TRUCKCONVOY-20
          TRUCKCONVOY-21      TRUCKCONVOY-22      TRUCKCONVOY-23
          TRUCKCONVOY-24      TRUCKCONVOY-25      TRUCKCONVOY-26
          TRUCKCONVOY-27      TRUCKCONVOY-28      TRUCKCONVOY-29
          TRUCKCONVOY-30      TANKTRUCKS-10       TANKTRUCKS-11
          TANKTRUCKS-12       TANKTRUCKS-13       TANKTRUCKS-14
          TRUCKCONVOY-31      TRUCKCONVOY-32      TRUCKCONVOY-33
          TRUCKCONVOY-34      Bonneyman-1         Altair-1
          Algol-2             Antares-3           Bellatrix-4
          Capella-5           Denebola-6          Pollux-7
          Regulus-8

==============================================================================
          COMMAND:  (S)ingle, (P)repo, (A)ll, (M)odify, (R)eturn
```

Figure 7 Transporter Listing

Name. It is separated from the Transporter Name by a dash. To view a single Transporter, press "**S**".

3. S3 will be asked for the Name of the Transporter that you wish to display. Type the Name exactly as it appears in the list above. Pay special attention to upper and lower case characters as S3 is case sensitive. When the Name of the Base or Unit has been typed, Press **<ENTER>**.

4. S3 will then ask for the Vehicle ID of the Transporter that you wish to display. Type the Vehicle ID as it appears in the list above. Do not include the dash which separates the Name from the Vehicle ID. When the Vehicle ID of the Transporter has been typed, Press **<ENTER>**.

5. The a Transporter display similar to Figure 8 will be presented on the screen. To return, press "**Y**" at the prompt.

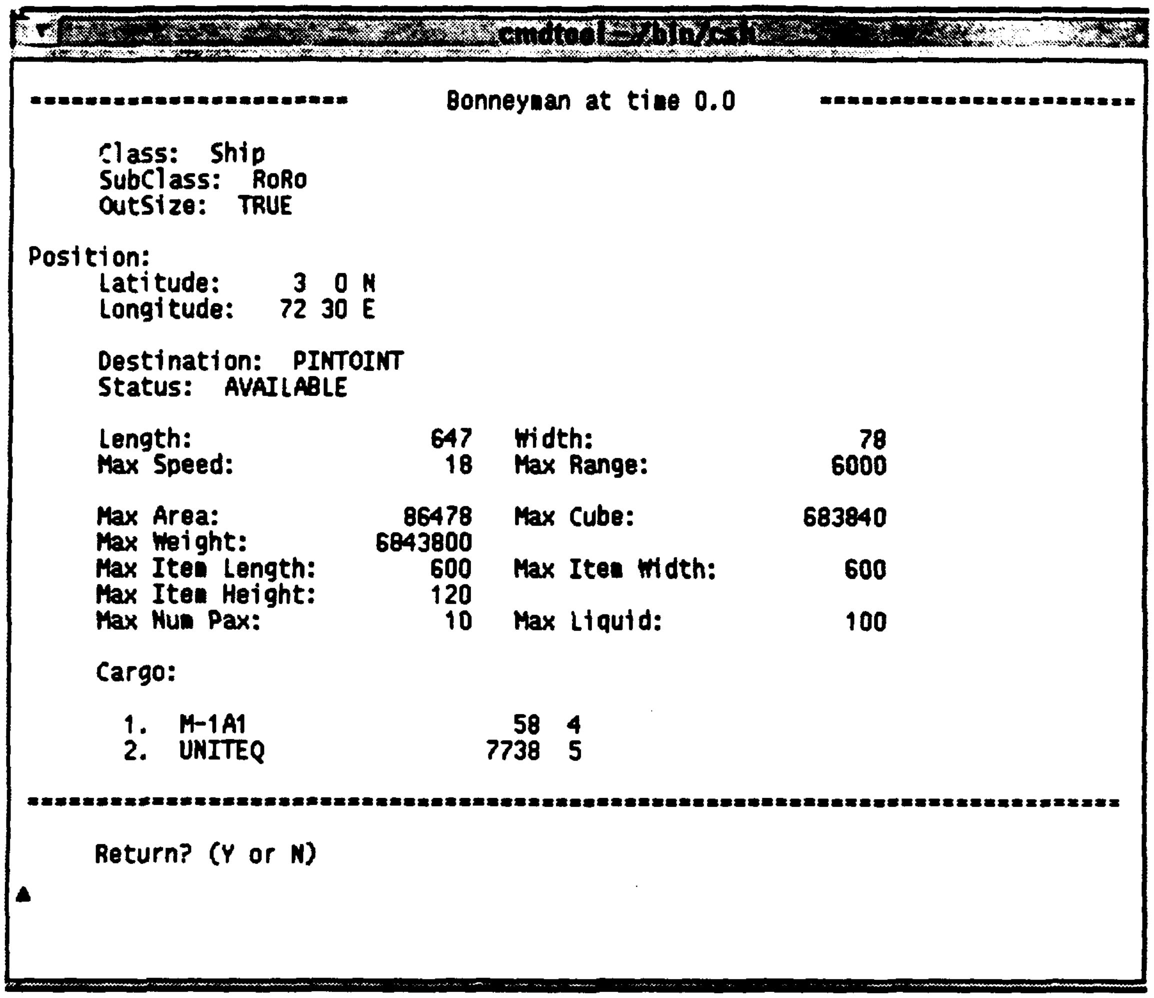
6. The list of the Transporters will reappear on the screen. When you wish to return to the previous Base listing, press "**Y**" at the prompt.

b. Viewing All Transporters In A Scenario

At times, it is helpful to display all of the Transporters in a Scenario sequentially. These occasions are very rare, but the feature was included at any rate. Since a typical scenario might involve hundreds of Transporters, this is not the recommended method for viewing Transporters. The display will be identical to the display of a single Transporter, however, all of the Transporters in the Scenario will be displayed in turn. If you wish to view all Scenario Transporters:

1. From the **Time Step Menu** press "**T**".

2. A list of all the Transporters in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 7. To view all Transporters, press "**A**".



```
======================        Bonneyman at time 0.0        ======================

     Class:  Ship
     SubClass:  RoRo
     OutSize:  TRUE

Position:
     Latitude:      3  0 N
     Longitude:    72 30 E

     Destination:  PINTOINT
     Status:  AVAILABLE

     Length:                 647    Width:                 78
     Max Speed:               18    Max Range:           6000

     Max Area:             86478    Max Cube:          683840
     Max Weight:         6843800
     Max Item Length:        600    Max Item Width:       600
     Max Item Height:        120
     Max Num Pax:             10    Max Liquid:           100

     Cargo:

       1.  M-1A1                   58  4
       2.  UNITEQ                7738  5

=============================================================================

     Return? (Y or N)
```

Figure 8 The Transporter Display

3. A display of the first Transporter on the list will appear on the screen. When you wish to view the next Transporter on the list, press **any key** to continue. If you wish to return to the previous Transporter listing, press "**Y**" when asked if you wish to return.

c. Activating A Prepositioned Transporter

Prepositioned Transporters are important in modeling the deployment of Units to the theater of operations. Before a Prepositioned Transporter can bring its Cargo to its assigned destination, it must be activated. This gives the user control over Prepo use to ensure coordination with game events and player decisions. To activate a Prepositioned Transporter:

1. From the **Time Step Menu** press "**T**".

2. A list of all the Transporters in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 7. To activate Prepositioned Transporters, press "**P**".

3. S3 will ask if you wish to activate a Prepositioned Transporter. To activate a Prepositioned Transporter, press "**A**"

4. A list of all the Prepositioned Transporters will be displayed. Enter the Name of the Prepositioned Transporter you wish to activate exactly as it appears on the screen and press **<ENTER>**.

5. S3 will then ask for the Vehicle ID of the Transporter that you wish to destroy. Type the Vehicle ID as it appeared in the listing. Do not include the dash which separates the Name from the Vehicle ID. When the Vehicle ID of the Transporter has been typed, Press **<ENTER>**.

NOTE

A Prepositioned Transporter may only be activated once per scenario execution. Activating a Prepo that has been previously activated will result in a runtime error.

d. *Destroying a Transporter*

To allow for the possibility of enemy action against a logistics system, S3 allows for the destruction of Transporters. When a Transporter is destroyed, it is removed from the scenario and its Cargo is destroyed. The corresponding Commodity Order levels at the receiving Bases are adjusted so that they may reorder the Commodity. To destroy a Transporter:

1. From the **Time Step Menu** press "**T**".

2. A list of all the Transporters in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 7. To destroy a Transporter press "**M**" for "Modify".

3. S3 will show another list of commands. To destroy a Transporter, press "**D**". S3 will then ask you to confirm your selection. Press "**Y**" to confirm that you wish to destroy a Transporter.

4. S3 then asks for the model Name of the Transporter you wish to destroy. Enter the Name of the Transporter you wish to destroy exactly as it appeared on the list of Transporters and press **<ENTER>**.

5. S3 will then ask for the Vehicle ID of the Transporter that you wish to destroy. Type the Vehicle ID as it appeared in the listing. Do not include the dash which separates the Name from the Vehicle ID. When the Vehicle ID of the Transporter has been typed, Press **<ENTER>**.

6. S3 will now remove the Transporter from the game. You will be returned to the previous list of commands. To return, press "**R**".

4. Displaying Commodities

Just like Bases, Units and Transporters, you can display Scenario Commodities. All Commodities in a Scenario may be displayed at once, or, a search may be conducted to locate the individual instances of a single Commodity. The amount On Hand, the amount On Order, the Stocking Objective, and the Consumption Rate will be displayed for every Base, Unit, and Transporter that stocks the Commodity or is carrying it as Cargo. The search is an extremely helpful tool when you want to know the theater-wide distribution of any particular Commodity.

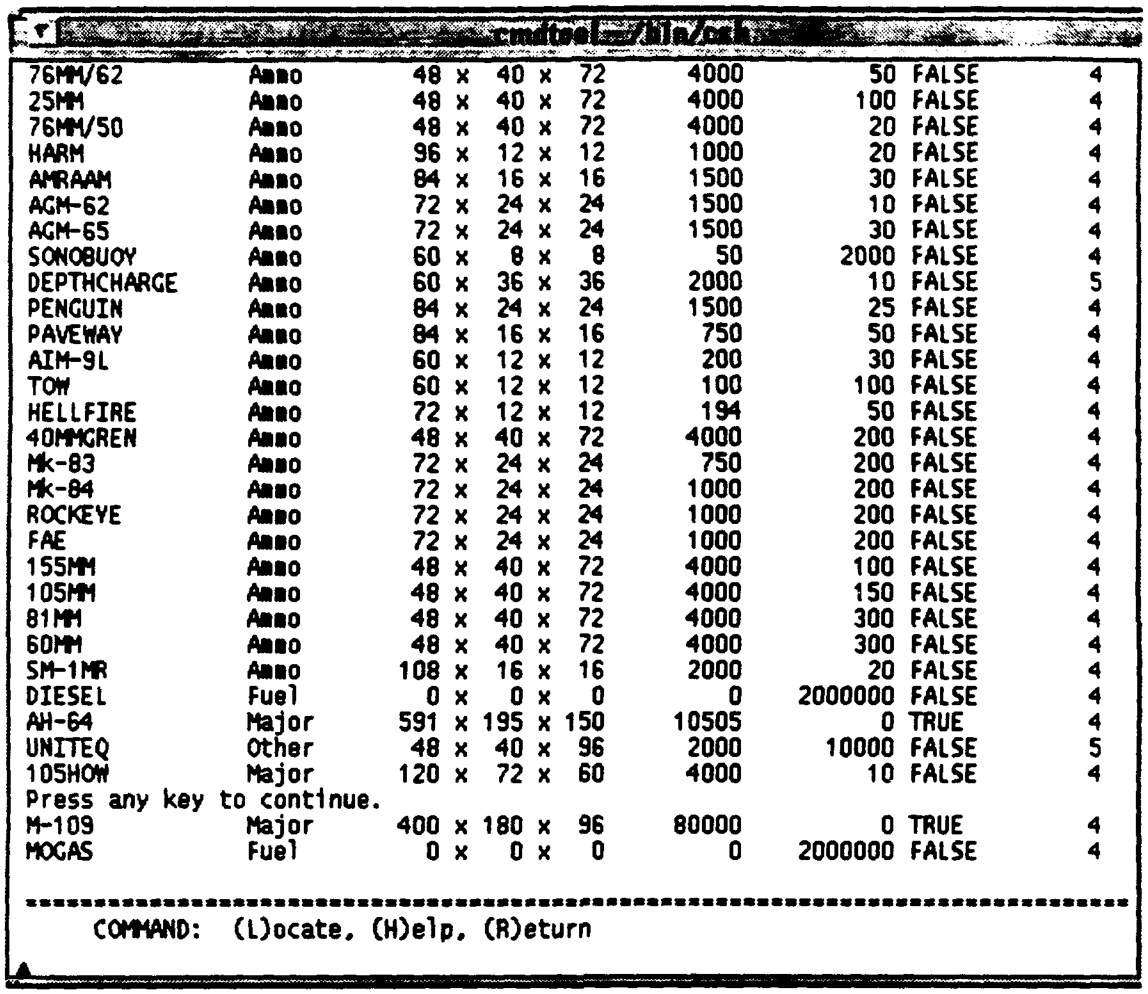
a. The Commodity Display

When you invoke the Commodity Display, a master listing of every Commodity in the Scenario will be presented on the screen. The Class, Dimensions, Weight, Production Rate, Oversize Status, and Priority will be displayed with the Name of each Commodity. To view the Commodity Display:

1. From the **Time Step Menu** press "C".
2. A list of all the Commodities in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 9. As the list of Commodities in a Scenario can be quite long, the Display will be presented one screen at a time. To view the next screen, merely press **any key**. To return, press "**R.**"

b. Locating A Commodity

When you locate a Commodity, you are searching every Base, Unit, and Transporter in the scenario for the Commodity selected. If the Commodity exists, the amount On Hand, the amount On Order, the Stocking Objective, and the current Consumption Rate will be displayed along with the Name of the Base, Unit or Transporter that is holding the



```
76MM/62        Ammo     48 x  40 x  72      4000        50 FALSE         4
25MM           Ammo     48 x  40 x  72      4000       100 FALSE         4
76MM/50        Ammo     48 x  40 x  72      4000        20 FALSE         4
HARM           Ammo     96 x  12 x  12      1000        20 FALSE         4
AMRAAM         Ammo     84 x  16 x  16      1500        30 FALSE         4
AGM-62         Ammo     72 x  24 x  24      1500        10 FALSE         4
AGM-65         Ammo     72 x  24 x  24      1500        30 FALSE         4
SONOBUOY       Ammo     60 x   8 x   8        50      2000 FALSE         4
DEPTHCHARGE    Ammo     60 x  36 x  36      2000        10 FALSE         5
PENGUIN        Ammo     84 x  24 x  24      1500        25 FALSE         4
PAVEWAY        Ammo     84 x  16 x  16       750        50 FALSE         4
AIM-9L         Ammo     60 x  12 x  12       200        30 FALSE         4
TOW            Ammo     60 x  12 x  12       100       100 FALSE         4
HELLFIRE       Ammo     72 x  12 x  12       194        50 FALSE         4
40MMGREN       Ammo     48 x  40 x  72      4000       200 FALSE         4
Mk-83          Ammo     72 x  24 x  24       750       200 FALSE         4
Mk-84          Ammo     72 x  24 x  24      1000       200 FALSE         4
ROCKEYE        Ammo     72 x  24 x  24      1000       200 FALSE         4
FAE            Ammo     72 x  24 x  24      1000       200 FALSE         4
155MM          Ammo     48 x  40 x  72      4000       100 FALSE         4
105MM          Ammo     48 x  40 x  72      4000       150 FALSE         4
81MM           Ammo     48 x  40 x  72      4000       300 FALSE         4
60MM           Ammo     48 x  40 x  72      4000       300 FALSE         4
SM-1MR         Ammo    108 x  16 x  16      2000        20 FALSE         4
DIESEL         Fuel      0 x   0 x   0         0   2000000 FALSE         4
AH-64          Major   591 x 195 x 150     10505         0 TRUE          4
UNITEQ         Other    48 x  40 x  96      2000     10000 FALSE         5
105HOW         Major   120 x  72 x  60      4000        10 FALSE         4
Press any key to continue.
M-109          Major   400 x 180 x  96     80000         0 TRUE          4
MOGAS          Fuel      0 x   0 x   0         0   2000000 FALSE         4

==============================================================================
     COMMAND:  (L)ocate, (H)elp, (R)eturn
```

Figure 9 Commodity Lisitng

Commodity. To locate a Commodity:

1. From the **Time Step Menu** press "C".

2. A list of all the Commodities in the Scenario will be displayed with a command prompt. This list will be similar to the list in Figure 9. As the list of Commodities in a Scenario can get quite long, the Display will be presented one screen at a time. To view the next screen, merely press **any key**. To locate a Commodity, press "**L**".

3. S3 will then ask for the Name of the Commodity you wish to locate. Type the Commodity Name exactly as it appears in the listing above and press **<ENTER>**.

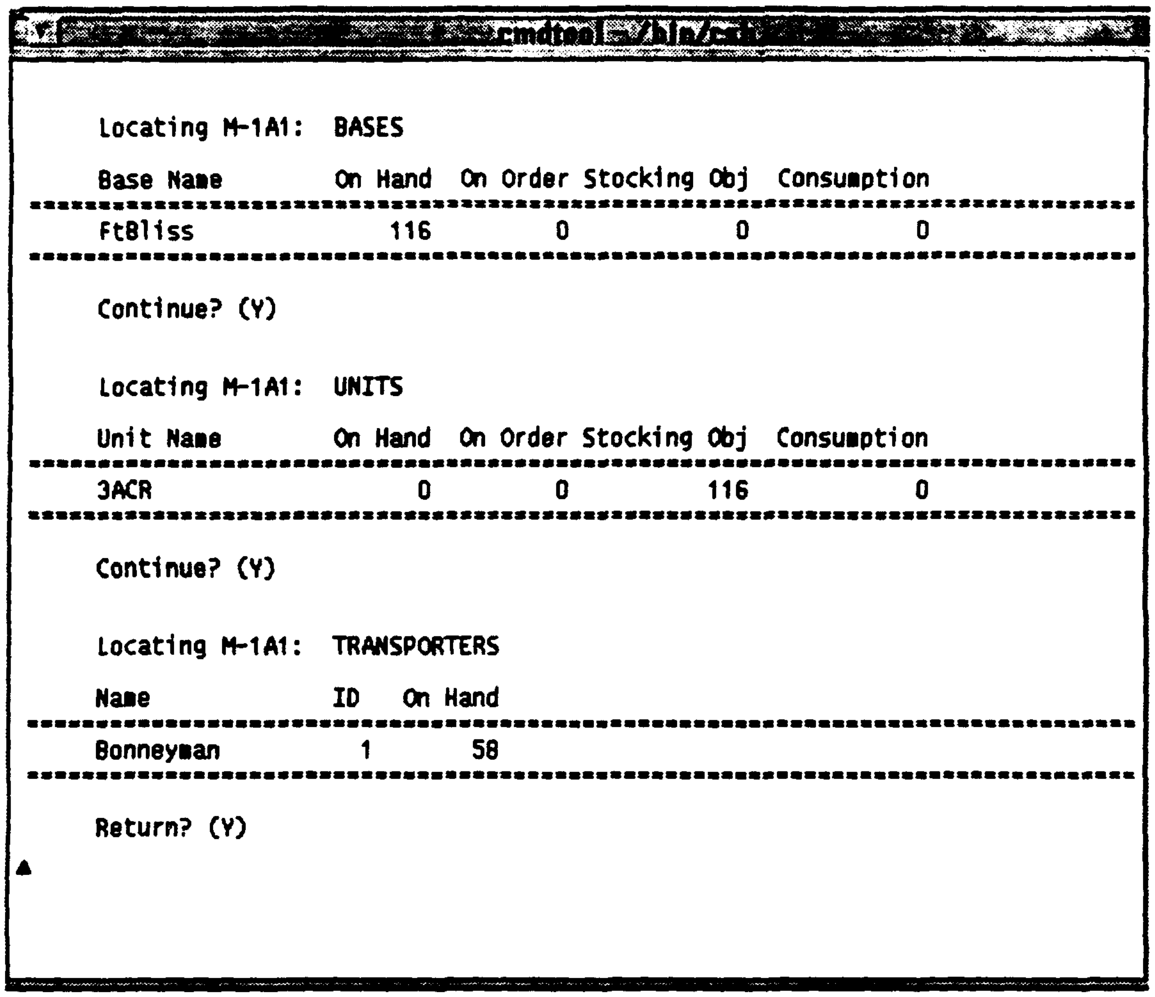
4. S3 will display the list of all Bases that stock the selected Commodity (Figure 10). You are given the option of continuing the search or returning to the Commodity Display. To view the list of Units and Transporters, press **any key**. To quit the search and return, press "**N**".

5. If you opt to continue, the list of Units stocking the selected Commodity will be displayed. Again you are given the option of quitting the search or continuing to view the Transporters. To view the list of Transporters, press **any key**. To quit the search, press "**N**"

6. If you opt to continue, the list of Transporters carrying the selected Commodity will be displayed. Here you are asked if you wish to return to the Commodity Display. By pressing **any key**, you will be returned to the Commodity Display. If you press "**N**", the entire search will be performed again.

5. Displaying Unit Supply Status

Whenever you enter the Time Step Menu, a graphic representation depicting the aggregate supply status for all Units in the scenario is displayed. This aggregate supply status is broken down by category and is based on the proportion of items in stock at each Unit. For example, if a Units stocks a total of 3000 vehicles of all types and has 1500 on hand, the Unit will contribute this proportion to the



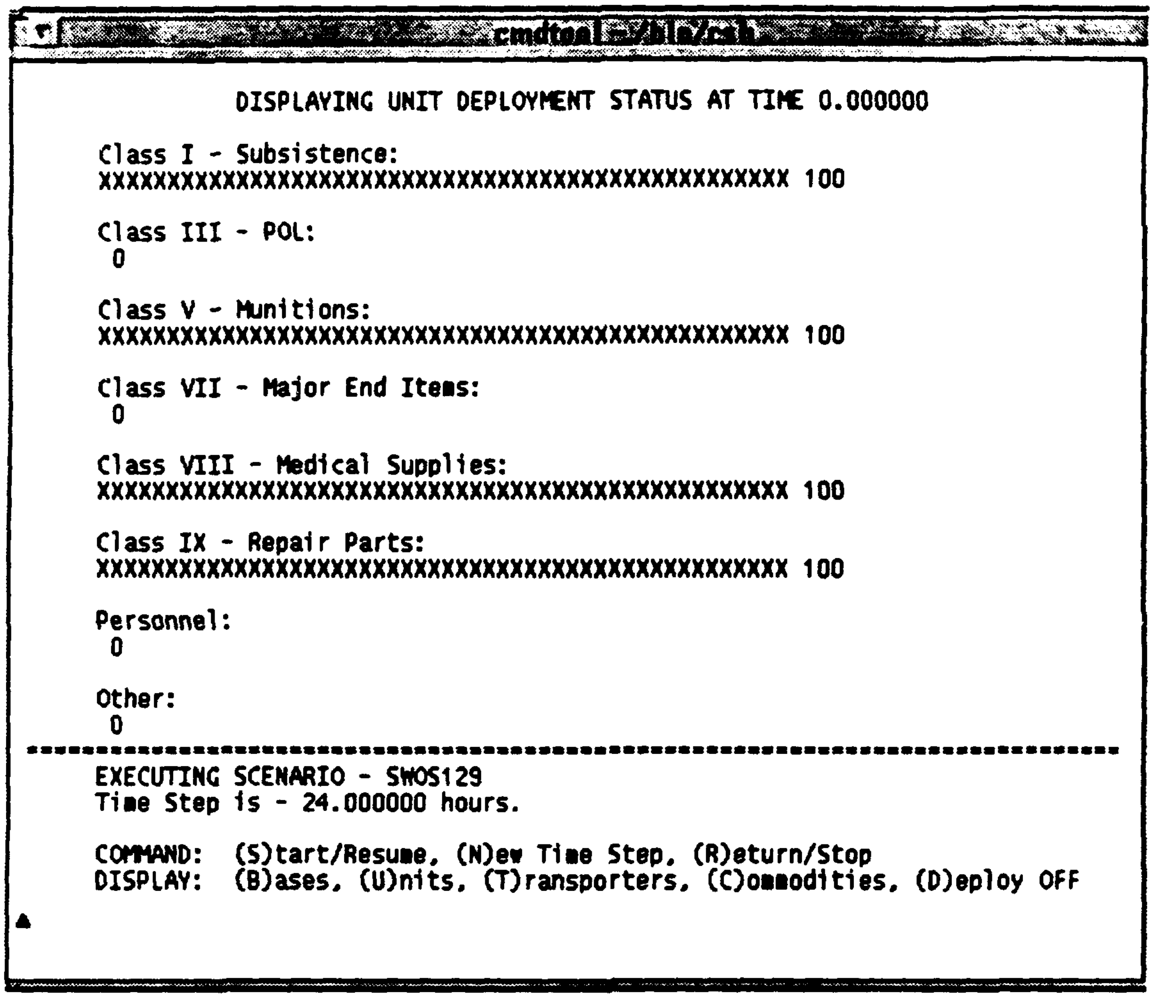
```
cmdtool - /bin/csh

    Locating M-1A1:  BASES

    Base Name         On Hand  On Order Stocking Obj  Consumption
==========================================================================
    FtBliss              116         0            0             0
==========================================================================

    Continue? (Y)

    Locating M-1A1:  UNITS

    Unit Name         On Hand  On Order Stocking Obj  Consumption
==========================================================================
    3ACR                   0         0          116             0
==========================================================================

    Continue? (Y)

    Locating M-1A1:  TRANSPORTERS

    Name              ID   On Hand
==========================================================================
    Bonneyman          1        58
==========================================================================

    Return? (Y)
▲
```

Figure 10 Commodity Location Display

aggregate percentage. In other words, the display counts numbers of Inventory line items, not weight or cube. By this measure a TOW II missile counts as much as a barrel of JP-5.

The same display is also available for all Deployment Commodities. When this status report is displayed, only those Commodities flagged as Deployment Commodities for each Unit will be considered. This display was devised to provide a visual representation of the progress of Unit deployment to the theater of operations. To toggle between each display:

1. From the **Time Step Menu** press "**D**".
2. A display will automatically switch from Unit Supply Status to Unit Deployment Status (Figure 11).
3. To switch back to Unit Supply Status, press "**D**" again. This function is essentially a toggle switch for the display. The Time Step Menu will show whichever display is currently selected.



```
                 DISPLAYING UNIT DEPLOYMENT STATUS AT TIME 0.000000

     Class I - Subsistence:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Class III - POL:
      0

     Class V - Munitions:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Class VII - Major End Items:
      0

     Class VIII - Medical Supplies:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Class IX - Repair Parts:
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 100

     Personnel:
      0

     Other:
      0
********************************************************************************
     EXECUTING SCENARIO - SWOS129
     Time Step is - 24.000000 hours.

     COMMAND:  (S)tart/Resume, (N)ew Time Step, (R)eturn/Stop
     DISPLAY:  (B)ases, (U)nits, (T)ransporters, (C)ommodities, (D)eploy OFF
▲
```

Figure 11 Unit Deployment Status

III. BUILDING A SCENARIO

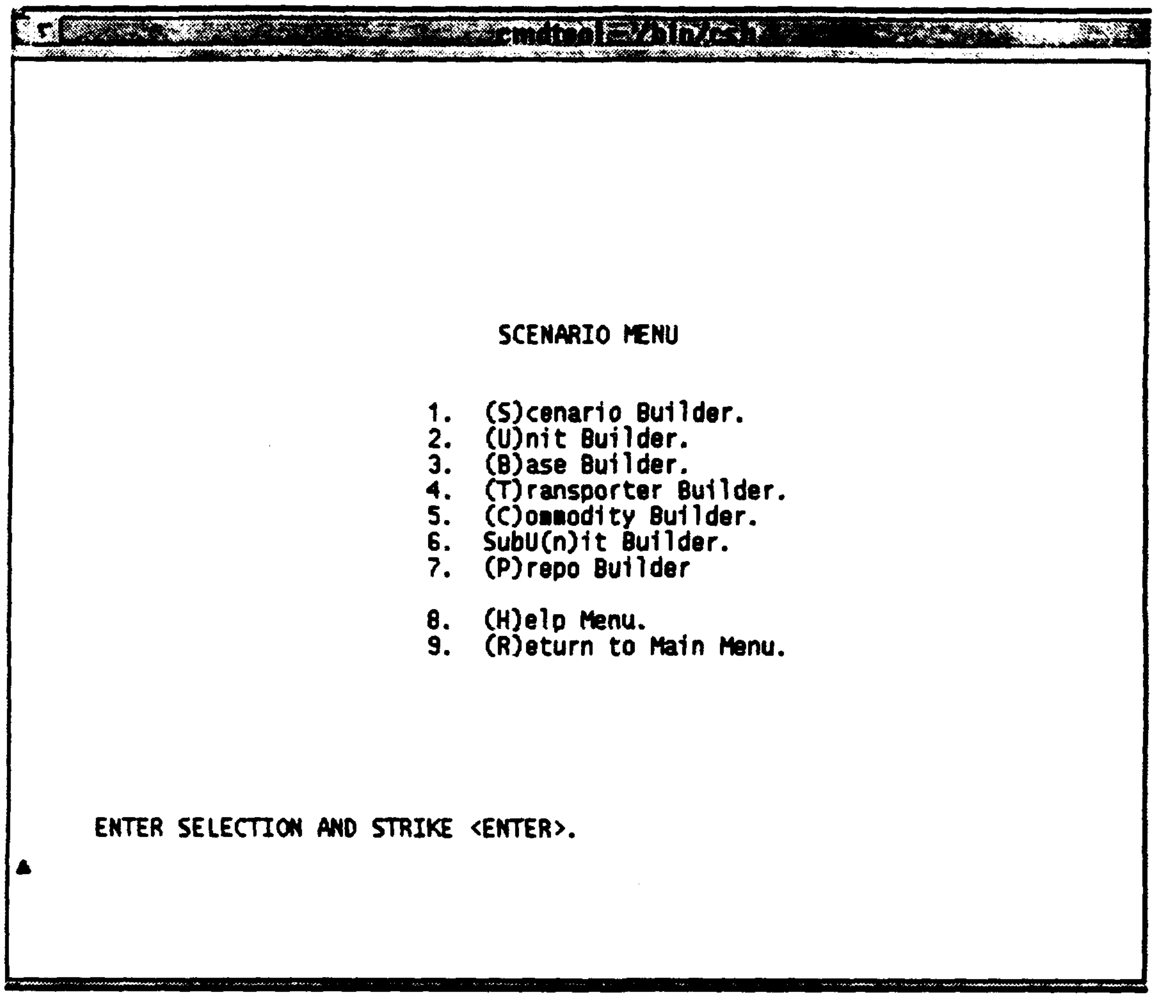
The other half of the S3 menu system is the Scenario Editor. Here, you can build all of the entities required to run the most complicated of Scenarios. When constructing a Scenario, there is a specific order in which to proceed. To avoid the necessity of reconstructing and modifying Bases and Units, it is important to follow this construction order closely:

- Commodities
- Transporters
- SubUnits
- Units
- Bases
- Prepositioned Transporters
- Scenarios

Of course, no harm will result to the program if this order is not maintained. This sequence is the most logical way to proceed given that later scenario entities are the products of those that have been previously constructed.

Construction of Scenarios begins when you enter the Scenario Menu from the Main Menu. To enter the Scenario Menu press "1" or "S" when in the **Main Menu**. There will be a momentary delay while S3 generates all the previously constructed Bases, Units, Transporters and Commodities. The Scenario Menu (Figure 12) will then be displayed.

The Scenario Menu is an enumerated menu much like the Main Menu. From here, you may construct all of the entities required to generate a scenario. Additionally, you may also modify any of the entities that have been previously



```
cmdtool - /bin/csh

                         SCENARIO MENU

                    1.   (S)cenario Builder.
                    2.   (U)nit Builder.
                    3.   (B)ase Builder.
                    4.   (T)ransporter Builder.
                    5.   (C)ommodity Builder.
                    6.   SubU(n)it Builder.
                    7.   (P)repo Builder

                    8.   (H)elp Menu.
                    9.   (R)eturn to Main Menu.

   ENTER SELECTION AND STRIKE <ENTER>.
```

Figure 12 The Scenario Menu

constructed. Unlike modifications made during execution, modifications made to entities while in the Scenario Editor are permanent.

A. BUILDING COMMODITIES

Commodities are the at heart of Scenario generation. They are required for the construction of SubUnits, Units, Bases, and Prepositioned Transporters. From the Scenario Menu you may either construct an entirely new Commodity or you may modify an existing Commodity.

1. Building a New Commodity

To build a new Commodity:

1. From the **Scenario Menu** press "**5**" or "**C**".
2. The Commodity Builder Menu will be displayed with a command prompt. To build a new Commodity, press "**1**" or "**N**".
3. S3 will then ask for the Name of the Commodity that you wish to construct. Type the new Commodity Name and press **<ENTER>**. Try to keep the Name of the Commodity to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.
4. Next, S3 will ask for the Class of the Commodity that you wish construct. A list of Classes will be displayed for you to choose from. Press the appropriate key to enter the Commodity Class.
5. S3 will ask for the Commodity Production Rate, next. This real number is the amount of the Commodity that will be produced by the Supply source each simulated day. Enter the Production Rate in the format **X.X** and press **<ENTER>**.
6. S3 will then ask for the Commodity Dimensions, one value at a time. Input the Length, Width, Height and Weight in the **X.X** format when prompted. All dimensions to be entered in inches and weight is to be entered in pounds.

7. Next you will be asked to enter the new Commodity's Priority. This Priority will be the Priority that the Commodity will take during normal circumstances. Priority is always a integer between 1 and 12. Type the pricrity in the format **XX** and press **<ENTER>**.

8. S3 now asks for the Commodity Emergency Priority. This Priority will be the Priority that the Commodity will take during emergency circumstances. Emergency circumstances are defined when building a Unit or Base Inventory. Specifically, orders for a Commodity whose On Hand amount has dropped below its Emergency Order Point will be processed at the Emergency Priority. Emergency Priority is always a integer between 1 and 12 and is normally lower than the normal Priority, although there are no restrictions. Type the priority in the format **XX** and press **<ENTER>**.

9. Construction of the Commodity is now complete. S3 will display the newly constructed Commodity for your approval. If you wish to save the new Commodity, press "**Y**" at the prompt. If not, press "**N.**"

2. Modifying an Existing Commodity

Modification of a previously constructed Commodity is very similar to its construction. To modify a Commodity:

1. From the **Scenario Menu** press "**5**" or "**C**".

2. The Commodity Builder Menu will be displayed with a command prompt. To modify an existing Commodity, press "**2**" or "**E**".

3. S3 will then ask for the Name of the Commodity that you wish to modify. Type the Commodity Name exactly and press **<ENTER>**.

4. S3 will display the Commodity and a list of modifications that you may make. You may modify the Name, Class, Dimensions, Weight, Production Rate and Priorities. Press the appropriate key to modify a Commodity characteristic.

5. If you pressed "**N**" to modify the Commodity Name, S3 will ask you to input the new Name of the Commodity. Type the new Commodity Name and press **<ENTER>**. Try to keep the Name of the Commodity to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.

6. If you pressed "**C**" to modify the Commodity Class, S3 will ask you to input the new Class of the Commodity that you wish to modify. A list of Classes will be displayed for you to choose from. Press the appropriate key to enter the Commodity Class.

7. If you pressed "**D**" to modify the Dimensions, S3 will ask for the Commodity Dimensions, one value at a time. Input the Length, Width, and Height in the **X.X** format when prompted. All dimensions to be entered in inches.

8. If you pressed "**W**" to modify the Commodity Weight, S3 will ask for the Commodity Weight in pounds. Input the Weight in the **X.X** format when prompted.

9. If you pressed "**T**" to modify the Commodity Production Rate, S3 will ask you to input the new Production Rate. Enter the Production Rate in the format **X.X** and press **<ENTER>**.

10. If you pressed "**P**" to modify the Commodity Priority, you will be asked which Priority to modify. You may modify either the Emergency or Normal Priority. Press the appropriate key and enter the new Commodity Priority when prompted. Priority is always a integer between 1 and 12. Type the Priority in the format **XX** and press **<ENTER>**.

B. BUILDING TRANSPORTERS

The next step in the construction of a scenario is building the different Transporters that are to be modeled.

1. Building a New Transporter

To construct a Transporter:

1. From the **Scenario Menu** press "**4**" or "**T**".

2. The Transporter Builder Menu will be displayed with a command prompt. To build a new Transporter, press "**1**" or "**N**".

3. S3 will then ask for the Name of the Transporter that you wish to construct. Type the new Transporter Name and press **<ENTER>**. Try to keep the Name to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.

4. Next, S3 will ask for the Class of the Transporter. A list of Classes will be displayed for you to choose from. Press the appropriate key to enter the Transporter Class.

5. S3 will then ask for the SubClass of the Transporter. A list of SubClasses will be displayed for you to choose from. Press the appropriate key to enter the Transporter SubClass.

6. S3 will ask for the Transporter Maximum Speed in real nautical miles per hour. Enter the Maximum Speed in the format, **X.X,** and press **<ENTER>**.

7. S3 will ask for the Transporter Maximum Range in real nautical miles. Enter the Maximum Range in the format, **X.X,** and press **<ENTER>**.

8. S3 will ask for the Transporter Dimensions one value at a time. These dimensions are to be entered in real feet. Enter each value in the format, **X.X,** and press **<ENTER>**.

9. S3 will ask for the Transporter Cargo Area in real square feet. Enter the value in the format, **X.X,** and press **<ENTER>**.

10. S3 will ask for the Transporter Cargo Cube in real cubic feet. Enter the value in the format, **X.X,** and press **<ENTER>**.

11. S3 will ask for the Transporter Cargo Dimensions one value at a time. These dimensions are to be entered in real inches. Enter each value in the format, **X.X,** and press **<ENTER>**.

12. S3 will ask for the Transporter Passenger Capacity. Enter the number of passengers the Transporter may carry in the format, **X.X,** and press **<ENTER>**.

13. S3 will ask for the Transporter Liquid Capacity. Enter the number of barrels of liquid the Transporter may carry in the format, **X.X,** and press **<ENTER>**.

14. Construction of the Transporter is now complete. S3 will display the newly constructed Transporter for your approval. If you wish to save the new Transporter, press "**Y**" at the prompt. If not, press "**N**."

2. Modifying an Existing Transporter

Modification of a previously constructed Transporter is very similar to its construction. To modify a Transporter:

1. From the **Scenario Menu** press "**4**" or "**T**".

2. The Transporter Builder Menu will be displayed with a command prompt. To modify an existing Transporter, press "**2**" or "**E**".

3. S3 will then ask for the Name of the Transporter that you wish to modify. Type the Transporter Name exactly and press **<ENTER>**.

4. S3 will display the Transporter and a list of modifications that you may make. You may modify the Name, Dimensions, Performance, and Cargo Parameters. Press the appropriate key to modify a Transporter characteristic.

5. If you pressed "**N**" to modify the Transporter Name, S3 will ask you to input the new Name of the Transporter. Type the new Name and press **<ENTER>**. Try to keep the Name of the Transporter to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.

6. If you pressed "**D**" to modify the Transporter Dimensions, S3 will ask for the Dimensions, one value at a time. Input the Length and Width in the **X.X** format when prompted. All dimensions to be entered in feet.

7. If you pressed "**P**" to modify the Transporter Performance, S3 will ask for the Transporter Maximum Speed and Range. Input the values in the **X.X** format when prompted.

8. If you pressed "**G**" to modify the Transporter Dimensions, S3 will ask for the Cargo Dimensions, Cube and Area, one value at a time. Input the Length, Width, and Height, Cube and Area in the **X.X** format when prompted.

C. BUILDING SUBUNITS

SubUnits are critical to the construction of Units. Without SubUnits, constructing Units might become a difficult and tedious task. To construct a SubUnit, all of the Commodities by which the SubUnit will be defined must have been constructed.

1. Building a New SubUnit

To construct a new SubUnit:

1. From the **Scenario Menu** press "**6**" or "**N**".

2. The SubUnit Builder Menu will be displayed with a command prompt. To build a new SubUnit, press "**1**" or "**N**".

3. S3 will then ask for the Name of the SubUnit that you wish to construct. Type the new SubUnit Name and press **<ENTER>**. Try to keep the Name to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.

4. Next, S3 will ask for the type of the SubUnit. The list of choices will be displayed from which to select. Press the appropriate key to enter the SubUnit type.

5. A list of all the previously built Commodities will be displayed. S3 will then ask if you wish to add one of these Commodities. The Commodities that are entered in the SubUnit construction process represent the Commodities that are normally associated with the operation of a SubUnit. The consumption rates of these Commodities will be aggregated when a Unit is constructed from a number of SubUnits. Press the "**Y**" key if you wish to enter a Commodity.

6. If you have decided to enter a Commodity, S3 will ask for the Commodity's Name. Type the Commodity's Name exactly and press **<ENTER>**.

7. S3 will then ask you to enter the Consumption Rates of the Commodity at the different Combat Intensity levels. These Consumption Rates are defined in units of the Commodity consumed per simulated day. When prompted, enter values in the format, **X.X,** and press **<ENTER>**.

8. S3 will ask if you want the Commodity to count towards deployment. If so, press "**Y**". If not, press "**N**".

9. The Commodity is now added to the SubUnit Inventory. S3 will display the SubUnit and return to Step 5, for you to add another Commodity.

10. Once you are satisfied with the SubUnit, decline to add another Commodity. The SubUnit will be saved to a data file.

2. Modifying an Existing SubUnit

To modify an existing SubUnit:

1. From the **Scenario Menu** press "**6**" or "**N**".

2. The SubUnit Builder Menu will be displayed with a command prompt. To modify an existing SubUnit, press "**2**" or "**E**".

3. S3 will ask which type of SubUnit you wish to modify. Press the appropriate key at the prompt.

4. A display of all of the SubUnits of the selected type will be presented. S3 will then ask for the Name of the SubUnit that you wish to modify. Type the SubUnit Name exactly as it appears and press **<ENTER>**.

5. Since SubUnits are merely a listing of Commodities, S3 gives a choice of adding a new Commodity or modifying an existing one. Press the appropriate key to select the method of modifying the SubUnit.

6. If you pressed "**A**" to add a new Commodity proceed as outlined in Steps 5 to 8 of Section III.C.1 **Building A SubUnit**.

7. If you pressed "**M**" to modify a Commodity already in the SubUnit Inventory, you will be asked to enter the Commodity Name. Type the Commodity Name exactly and press **<ENTER>**.

    a. A list will be displayed from which you can choose the desired modification. Press the appropriate key to select the method of modifying the SubUnit.

    b. If you pressed "**S**" to modify the Commodity Stocking Objective, S3 will ask you to enter the new Stocking Objective. Type the value in the **X.X** format and press **<ENTER>**.

    c. If you pressed "**D**" to select if the Commodity will count towards deployment, press the appropriate key when prompted.

d. If you pressed "**C**" to modify the Commodity Consumption Rates, S3 will ask you to enter the new Consumption Rates one by one. When prompted, type the value in the **X.X** format and press **<ENTER>**.

8. The SubUnit will be redisplayed and you will be given the opportunity to make further modifications. To return to the SubUnit Builder Menu, press "**R.**"

D. BUILDING UNITS

Units may be created after you are satisfied that there are sufficient SubUnits, Transporters, and Bases already constructed.

1. Building a New Unit

To build a Unit:

1. From the **Scenario Menu** press "**2**" or "**U**".
2. The Unit Builder Menu will be displayed with a command prompt. To build a new Unit, press "**1**" or "**N**".
3. S3 will then ask for the Name of the Unit that you wish to construct. Type the new Unit Name and press **<ENTER>**. Try to keep the Name to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.
4. Next, S3 will ask if you wish the Unit to delay its activation. If you press "**Y**", you will be asked to enter the number of days from the beginning of the scenario that you wish the Unit to activate. Once the Unit activates, it will begin to monitor its Inventory and mobilize if it is not already "In Place." If you are not sure when the Unit will need to activate, set the delay sufficiently far in the future and activate manually during the execution.
5. You will then be asked if the Unit type is Air, Land, or Sea. Press the appropriate key at the prompt.
6. S3 will ask you to enter the Unit's Position. Enter the Latitude and Longitude in the format **DDD MM C** and

press **<ENTER>**.

7. You will then be asked a series of questions concerning the existence of Ports at the Unit. If the Unit has the indicated Port, press "**Y**." If not, press "**N**."

8. Each of the Ports you selected in the previous step are now constructed. S3 will ask you to enter the Maximum Capacity and Maximum Vehicle Size for each. Enter the values requested in the **XX** format and press **<ENTER>**.

9. S3 will then ask if you want to add Transporters. If you press "**Y**," a list of all the Transporters will be displayed. Input the Name of the Transporter that you wish to add to the Unit.

10. You will then be asked to give the number of the selected Transporter to add to the Unit. Type the number in the **XX** format and press **<ENTER>**.

11. Repeat Steps 9 and 10 until you are satisfied with the numbers and types of Transporters available at the Unit.

12. S3 will then ask if you want to add SubUnits. SubUnit selection will depend on Unit type. If you opt to add SubUnits, the appropriate list of SubUnits will be displayed.

13. Input the Name of a SubUnit when prompted by S3.

14. S3 will then ask for the number of the selected SubUnit that are attached to the Unit. Input the number in the **X.X** format and press **<ENTER>**.

15. Repeat Steps 12 through 14 until you are satisfied with the numbers and types of SubUnits comprising the Unit. S3 will ask if you are finished adding SubUnits. If you are finished, press "**Y**," otherwise press "**N**."

16. S3 will display the current Inventory for the Unit and ask if you wish to add any other Commodities not added by the SubUnit selection process. If so, the list of constructed Commodities will be displayed and S3 will ask for a Commodity Name. Type the desired Commodity Name exactly and press **<ENTER>**.

17. S3 will then ask you to enter the amount On Hand, the Stocking Objective and the Order Point for the

selected Commodity. When prompted, enter values in the format, **X.X,** and press **<ENTER>**.

18. Repeat Steps 16 and 17 until you are satisfied with Unit Inventory.

19. The Unit construction process is essentially complete. The Unit will be displayed and you will given the opportunity to make modifications before saving.

2. Modifying an Existing Unit

To modify a Unit:

1. From the Scenario Menu press "**2**" or "**U**".

2. The Unit Builder Menu will be displayed with a command prompt. To modify an existing Unit, press "**2**" or "**E**".

3. S3 will then ask for the Name of the Unit that you wish to modify. Type the Unit Name exactly and press **<ENTER>**.

4. Modification is now identical to modification during execution. Refer to Sections II.B.3 and II.B.4.

E. BUILDING BASES

1. Building a New Base

Building a Base is very similar to building a Unit with a few exceptions. First, Bases must be categorized in groups. Second, Bases do not use SubUnits to build their Inventories. To build a Base:

1. From the **Scenario Menu** press "**3**" or "**B**".

2. The Base Builder Menu will be displayed with a command prompt. To build a new Base, press "**1**" or "**N**".

3. S3 will then ask for the Name of the Base that you wish to construct. Type the new Base Name and press **<ENTER>**. Try to keep the Name to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.

4. Next, S3 will ask for the group in which you wish the new Base to be placed. The choices are CONUS, ILOC and Theater. Press the appropriate key to select the Base's group.

5. S3 will ask you to enter the Base's Position. Enter the Latitude and Longitude in the format **DDD MM C** and press **<ENTER>**.

6. You will then be asked a series of questions concerning the existence of Ports at the Base. If the Base has the indicated Port, press "**Y**." IF not, press "**N**."

7. Each of the Ports you selected in the previous step are now constructed. S3 will ask you to enter the Maximum Capacity and Maximum Vehicle Size for each. Enter the values requested in the **XX** format and press **<ENTER>**.

8. S3 will then ask if you want to add Transporters. If you press "**Y**," a list of all the Transporters will be displayed. Input the Name of the Transporter that you wish to add to the Base.

9. You will then be asked to give the number of the selected Transporter to add to the Base. Type the number in the **XX** format and press **<ENTER>**.

10. Repeat Steps 8 and 9 until you are satisfied with the numbers and types of Transporters available at the Base.

11. S3 will then ask if you wish to add Commodities to the Base Inventory. If so, the list of constructed Commodities will be displayed and S3 will ask for a Commodity Name. Type the desired Commodity Name exactly and press **<ENTER>**.

12. S3 will then ask you to enter the amount On Hand, the Stocking Objective and the Order Point for the selected Commodity. When prompted, enter values in the format, **X.X,** and press **<ENTER>**.

13. Repeat Steps 11 and 12 until you are satisfied with Base Inventory.

14. The Base construction process is essentially complete. The Base will be displayed and you will given the opportunity to make modifications before saving.

2. Modifying an Existing Base

To modify a Base:

1. From the Scenario Menu press "**3**" or "**B**".
2. The Base Builder Menu will be displayed with a command prompt. To modify an existing Base, press "**2**" or "**E**".
3. S3 will then ask for the Name of the Base that you wish to modify. Type the Base Name exactly and press **<ENTER>**.
4. Modification is now identical to modification during execution. Refer to Sections II.B.3 and II.B.4.

F. BUILDING PREPOS

A Prepositioned Transporter is merely a previously constructed Transporter that has a specified Cargo, Position and destination. When activated, these Transporters will go directly to their destinations and unload their Cargo. The Cargo may continue on to other destinations based on a constructed route. After delivery of its Cargo, a Prepo makes itself available to the Transporter Manager for assignment.

1. Building a New Prepo

To build a Prepositioned Transporter:

1. From the **Scenario Menu** press "**7**" or "**P**".
2. The Prepo Builder Menu will be displayed with a command prompt. To build a new Prepo, press "**1**" or "**N**".
3. S3 will then ask for the Name of the Prepo that you wish to construct. Since a Prepo represents an individual vehicle rather than an entire class, the Name should be distinct. For example an FSS Prepo ship might be given its actual name (eg., Altair or Regulus). Type the new Prepo Name and press **<ENTER>**. Try to keep the Name to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.

4. Next, S3 will display the list of constructed Transporters. You are to select the Transporter model that the Prepo represents. Type the Transporter Name and press **<ENTER>**.

5. A list of Bases will be displayed from which you can choose the Cargo's ultimate destination. Type the Base Name, exactly as it appears, and press **<ENTER>**.

6. The current routing of the Cargo will be displayed. You will then be asked if you wish to enter a new intermediate destination for the Cargo. If so, a list of Bases will be displayed from which you can choose an intermediate destination. The first intermediate destination selected will be the destination of the Transporter. The Cargo will travel to any subsequent destinations before shipment to its ultimate destination. Type the Base Name, exactly as it appears, and press **<ENTER>**.

7. Repeat Step 6 until you are satisfied with the Cargo routing.

8. S3 will then ask if you want to add Commodities to the Cargo list. If you press "**Y**," a list of all the Commodities will be displayed. Input the Name of the Commodity that you wish to add to the Cargo.

9. You will then be asked to give the amount of the Commodity On Hand. Type the amount in the **X.X** format and press **<ENTER>**.

10. Repeat Steps 8 and 9 until you are satisfied with the numbers and types of Commodities manifested. When satisfied, decline to add another Commodity and the Prepo Construction process is complete.

2. Modifying an Existing Prepo

Version 1.0 does not include a method for modifying existing Prepositioned Transporters.

G. BUILDING SCENARIOS

Scenarios are essentially a list of Bases, Units and Prepos that interact in predefined ways. Besides picking the major entities in the Scenario you will define the origins of all Units and define the road and railroad networks that link Ports together.

1. Building a New Scenario

To build a Scenario:

1. From the **Scenario Menu** press "**1**" or "**S**".

2. The Scenario Builder Menu will be displayed with a command prompt. To build a new Scenario, press "**1**" or "**N**".

3. S3 will then ask for the Name of the Scenario that you wish to construct. Type the new Scenario Name and press **<ENTER>**. Try to keep the Name to fifteen characters or less. Otherwise, it will not display properly on certain screens. Also, remember, S3 is case sensitive.

4. Next, Scenario Bases are selected. A series of choices will be presented. You may add a Base to the Scenario, subtract a Base from the Scenario or view a list of the currently selected Bases. Press the appropriate key to make your selection.

   a. If you pressed "**A**," a list of Bases will be displayed from which you can choose. Type the Name of the Base you wish to add, exactly as it appears, and press **<ENTER>**.

   b. If you pressed "**S**," a list of previously selected Bases will be displayed from which you can choose. Type the Name of the Base you wish to remove, exactly as it appears, and press **<ENTER>**.

   c. If you pressed "**L**," a list of previously selected Bases will be displayed. Press **any key** to continue.

5. When you are satisfied with the Base selections, press "**R**" to return. S3 will move to the next step of the Scenario construction process.

6. Next, Scenario Units are selected. A series of choices will be presented. You may add a Unit to the Scenario, subtract a Unit from the Scenario or view a list of the currently selected Units. Press the appropriate key to make your selection.

a. If you pressed "**A**," a list of Units will be displayed from which you can choose. Type the Name of the Unit you wish to add, exactly as it appears, and press **<ENTER>**.

b. If you pressed "**S**," a list of previously selected Units will be displayed from which you can choose. Type the Name of the Unit you wish to remove, exactly as it appears, and press **<ENTER>**.

c. If you pressed "**L**," a list of previously selected Units will be displayed. Press **any key** to continue.

7. When you are satisfied with the Unit selections, press "**R**" to return. S3 will move to the next step of the Scenario construction process.

8. Next, Prepositioned Transporters selected. A series of choices will be presented. You may add a Prepo to the Scenario, subtract a Prepo from the Scenario or view a list of the currently selected Prepos. Press the appropriate key to make your selection.

   a. If you pressed "**A**," a list of Prepos will be displayed from which you can choose. Type the Name of the Prepo you wish to add, exactly as it appears, and press **<ENTER>**.

   b. If you pressed "**S**," a list of previously selected Prepos will be displayed from which you can choose. Type the Name of the Prepo you wish to remove, exactly as it appears, and press **<ENTER>**.

   c. If you pressed "**L**," a list of previously selected Prepos will be displayed. Press **any key** to continue.

9. When you are satisfied with the Prepo selections, press "**R**" to return. S3 will move to the next step of the Scenario construction process.

10. The next step is to provide all of the selected Units with Origins. An Origin is a Unit's home Base, from which all Deployment Commodities will come. The Logistics Manager also uses the Origin as the choice of last resort if there not an eligible supplier for a Unit that is "In Place." Normally, the Origin is a CONUS base that has an identical Inventory to that of its supported Unit. S3 will iterate through the Unit list asking for the Name of the Base you wish to be the Origin for each Unit. When prompted type the Name

of the Base, exactly, and press **<ENTER>**.

11. Next, S3 will then build the Scenario <u>railroad</u> network. S3 will iterate through the Base and Units lists. You will enter the Names of the Bases that can communicate with each Base or Unit. S3 will display the current Base name and a list of choices. You may add a Base to the network, subtract a Base from the network, view a list of the currently selected Bases or view a list of all the Scenario Bases and Units. Press the appropriate key to make your selection.

    a. If you pressed "**B**," a list of Scenario Bases and Units will be displayed. Press **any key** to continue.

    b. If you pressed "**N**," a list of previously selected network Bases and Units will be displayed. Press **any key** to continue.

    c. If you pressed "**A**," a list of eligible Bases and Units will be displayed from which you can choose. Type the Name of the Base or Unit you wish to add, exactly as it appears, and press **<ENTER>**.

    d. If you pressed "**S**," a list of previously selected Bases and Units will be displayed from which you can choose. Type the Name of the Base or Units that you wish to remove, exactly as it appears, and press **<ENTER>**.

12. When you are satisfied with the network selections, press "R" to return. S3 will move to the next step of the Scenario construction process.

13. Next, S3 will then build the Scenario <u>road</u> network. 3 will iterate through the Base and Units lists. You will enter the Names of the Bases that can communicate with each Base or Unit. S3 will display the current Base name and a list of choices. You may add a Base to the network, subtract a Base from the network, view a list of the currently selected Bases or view a list of all the Scenario Bases and Units. Press the appropriate key to make your selection.

    a. If you pressed "**B**," a list of Scenario Bases and Units will be displayed. Press **any key** to continue.

b. If you pressed "**N**," a list of previously selected network Bases and Units will be displayed. Press **any key** to continue.

c. If you pressed "**A**," a list of eligible Bases and Units will be displayed from which you can choose. Type the Name of the Base or Unit you wish to add, exactly as it appears, and press **<ENTER>**.

d. If you pressed "**S**," a list of previously selected Bases and Units will be displayed from which you can choose. Type the Name of the Base or Units that you wish to remove, exactly as it appears, and press **<ENTER>**.

14. When you are satisfied with the road network selections, press "R" to return. S3 will create the necessary Scenario data files and add the new Scenario to the Scenario List. You will be returned to the Scenario Builder Menu.

2. Modifying an Existing Scenario

To modify a Scenario:

1. From the **Scenario Menu** press "**1**" or "**S**".

2. The Scenario Builder Menu will be displayed with a command prompt. To modify an existing Scenario, press "**2**" or "**E**".

3. S3 will then ask for the Name of the Scenario that you wish to modify. Type the Scenario Name and press **<ENTER>**.

4. Here, the Scenario modification process is identical to building an new Scenario. Proceed as outlined in Steps 4 to 14 of Section III.G.1 **Building a New Scenario**.

APPENDIX J

SURGE AND SUSTAINMENT SIMULATION PROGRAM

```
DEFINITION MODULE Base;

{Base Object}

{Import statements}

FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;

FROM CommodQ IMPORT ALL CommodityClassType,
                    CommodityObj,
                    CommodityQObj;

FROM Node IMPORT NodeObj;

FROM Distant IMPORT PositionRecType;
FROM Trnsprt IMPORT TransporterObj,
                    TransporterQObj;
FROM Port IMPORT PortObj;

FROM Builder IMPORT BuilderObj;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;

{Type Declarations}

TYPE

BaseGroupType = (CONUS, ILOC, THEATER, UNIT);

BaseSubGroupType = (POE, POS, POD, NONE);


{========================================================================}
BaseObj = OBJECT(NodeObj)
{========================================================================}

{Fields}
     Group : BaseGroupType;
     SubGroup : BaseSubGroupType;

     BackOrders : ShipmentQObj;

     HasAirPort : BOOLEAN;
     AirPort : PortObj;

     HasSeaPort : BOOLEAN;
     SeaPort : PortObj;

     HasRail : BOOLEAN;
     RailYard : PortObj;

     HasTruckStop : BOOLEAN;
     TruckStop : PortObj;


{Methods}
     ASK METHOD SetGroup(IN NewGroup : STRING);
     ASK METHOD SetSubGroup(IN NewSubGroup : STRING);
```

```
      ASK METHOD SetHasAirPort(IN NewHasAirPort : STRING);
      ASK METHOD SetAirPort(INOUT NewAirPort : PortObj);

      ASK METHOD SetHasSeaPort(IN NewHasSeaPort : STRING);
      ASK METHOD SetSeaPort(INOUT NewSeaPort : PortObj);

      ASK METHOD SetHasRail(IN NewHasRail : STRING);
      ASK METHOD SetRailYard(INOUT NewRailYard : PortObj);

      ASK METHOD SetHasTruckStop(IN NewHasTruckStop : STRING);
      ASK METHOD SetTruckStop(INOUT NewTruckStop : PortObj);
      ASK METHOD OrderStuff(INOUT Item : CommodityObj);
      ASK METHOD FillOrder(INOUT Shipment : ShipmentObj);
      ASK METHOD ReceiveStuff(INOUT Cargo : ShipmentQObj);
      ASK METHOD BackOrderStuff(INOUT BackOrderShipment : ShipmentObj);
      ASK METHOD FillBackOrders;
      ASK METHOD RouteShipments(INOUT Shipment : ShipmentObj);
      TELL METHOD CheckInventory;

      ASK METHOD DumpFields;
      ASK METHOD Display;
      ASK METHOD Modify(INOUT builder : BuilderObj);
      ASK METHOD InputCommodities(INOUT builder : BuilderObj);
      ASK METHOD InputGroup;
      ASK METHOD InputSubGroup;

      ASK METHOD DisposePorts;

      OVERRIDE

      ASK METHOD ObjInit;
      ASK METHOD ObjTerminate;

  END OBJECT;

  {==========================================================================}
  BaseQObj = PROTO(MyQueueObj[NamedObj : #BaseObj])
  {==========================================================================}
  END PROTO;

  END MODULE.
```

```
IMPLEMENTATION MODULE Base;

{Base Object}

{Import statements}

FROM CommodQ IMPORT ALL CommodityClassType,
                        CommodityObj,
                        CommodityQObj;


FROM Node IMPORT NodeObj;
FROM Distant IMPORT PositionRecType,
                    OutputPosition,
                    InputPosition;
FROM Trnsprt IMPORT ALL TransporterClassType,
                        TransporterObj,
                        TransporterQObj;
FROM Port IMPORT PortObj;
FROM SimManager IMPORT StopTime;
FROM SimMod IMPORT SimTime;
FROM IOMod IMPORT ReadKey;
FROM WriteLine IMPORT WriteLine;
FROM CRTMod IMPORT ClearScreen;
FROM SOUTPUT IMPORT SOUTPUT;
FROM Builder IMPORT BuilderObj;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;
FROM LogMan IMPORT LogisticsManager;
FROM Builder IMPORT Builder;
FROM ScenEd IMPORT ScenarioEditor;
FROM TManage IMPORT TransporterManager;

{Definitions}

{============================================================================}
OBJECT BaseObj;
{============================================================================}

{METHODS}

{----------------------------------------------------------------------------}
 ASK METHOD SetGroup(IN NewGroup : STRING);
{----------------------------------------------------------------------------}

VAR

BEGIN

CASE NewGroup

WHEN "CONUS":
        Group := CONUS;

WHEN "ILOC":
        Group := ILOC;

WHEN "THEATER":
        Group := THEATER;
```

```
WHEN "UNIT":
        Group := UNIT;

OTHERWISE
        Group := THEATER;

END CASE;

END METHOD;
{-------------------------------------------------------------------------}
 ASK METHOD InputGroup;
{-------------------------------------------------------------------------}

VAR
CHR : CHAR;
BEGIN

LOOP
OUTPUT(     "Pick base group.  (C)ONUS, (I)LOC, (T)HEATER");
CHR := ReadKey();
IF (CHR = "C") OR (CHR = "c")
        SetGroup("CONUS");
        EXIT;
ELSIF (CHR = "I") OR (CHR = "i")
        SetGroup("ILOC");
        EXIT;
ELSIF (CHR = "T") OR (CHR = "t")
        SetGroup("THEATER");
        EXIT;
END IF;
END LOOP;

END METHOD:
{-------------------------------------------------------------------------}
 ASK METHOD SetSubGroup(IN NewSubGroup : STRING);
{-------------------------------------------------------------------------}

VAR

BEGIN

CASE NewSubGroup

WHEN "POE":
        SubGroup := POE;

WHEN "POS":
        SubGroup := POS;

WHEN "POD":
        SubGroup := POD;

WHEN "NONE":
        SubGroup := NONE;

OTHERWISE
        SubGroup := NONE;

END CASE;
END METHOD;

{---------------------------------------------------------------------------}
 ASK METHOD InputSubGroup;
{---------------------------------------------------------------------------}

VAR
CHR : CHAR;
BEGIN

LOOP
OUTPUT(     "Pick base group.  PO(E), PO(S), PO(D), (N)ONE");
CHR := ReadKey();
IF (CHR = "e") OR (CHR = "E")
        SetSubGroup("POE");
        EXIT;
ELSIF (CHR = "s") OR (CHR = "S")
        SetSubGroup("POS");
        EXIT;
ELSIF (CHR = "d") OR (CHR = "D")
        SetSubGroup("POD");
        EXIT;
ELSIF (CHR = "N") OR (CHR = "n")
        SetSubGroup("NONE");
        EXIT;
END IF;
END LOOP;

END METHOD;
{---------------------------------------------------------------------------}
 ASK METHOD SetHasAirPort(IN NewHasAirPort : STRING);
{---------------------------------------------------------------------------}

VAR

BEGIN

IF NewHasAirPort = "TRUE"
        HasAirPort := TRUE;
ELSE
        HasAirPort := FALSE;
END IF;

END METHOD;

{---------------------------------------------------------------------------}  AS
{---------------------------------------------------------------------------}

VAR

BEGIN

AirPort := CLONE(NewAirPort);

END METHOD;

{---------------------------------------------------------------------------}
 ASK METHOD SetHasSeaPort(IN NewHasSeaPort : STRING);
{---------------------------------------------------------------------------}
```

```
VAR

BEGIN

IF NewHasSeaPort = "TRUE"
        HasSeaPort := TRUE;
ELSE
        HasSeaPort := FALSE;
END IF;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetSeaPort(INOUT NewSeaPort : PortObj);
{---------------------------------------------------------------------------}

VAR

BEGIN

SeaPort := CLONE(NewSeaPort);

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD SetHasRail(IN NewHasRail : STRING);
{---------------------------------------------------------------------------}

VAR

BEGIN

IF NewHasRail = "TRUE"
        HasRail := TRUE;
ELSE
        HasRail := FALSE;
END IF;

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD SetHasTruckStop(IN NewHasTruckStop : STRING);
{---------------------------------------------------------------------------}

VAR

BEGIN

IF NewHasTruckStop = "TRUE"
        HasTruckStop := TRUE;
ELSE
        HasTruckStop := FALSE;
END IF;

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD SetRailYard(INOUT NewRailYard : PortObj);
{---------------------------------------------------------------------------}
```

```
VAR

BEGIN

RailYard := CLONE(NewRailYard);

END METHOD;

{-------------------------------------------------------------------------}
     ASK METHOD SetTruckStop(INOUT NewTruckStop : PortObj);
{-------------------------------------------------------------------------}

VAR

BEGIN

TruckStop := CLONE(NewTruckStop);

END METHOD;


{-------------------------------------------------------------------------}
     ASK METHOD OrderStuff(INOUT Item : CommodityObj);
{-------------------------------------------------------------------------}
     VAR
          ItemClass : CommodityClassType;

     BEGIN

OUTPUT("IN OrderStuff - ", Name);


         ASK LogisticsManager TO HandleBaseRequest(SELF,Item);


     END METHOD;

{-------------------------------------------------------------------------}
     ASK METHOD FillOrder(INOUT Shipment : ShipmentObj);
{-------------------------------------------------------------------------}

{method fills an order for a commodity either from the Logistics Manager or
backorder}

     VAR
          MyItem : CommodityObj;
          MyNewItem : CommodityObj;
          MyOnHand : REAL;
          MyStockTo : REAL;
          Receiver : BaseObj;
          ReceiverItem : CommodityObj;
          OrderQty : REAL;
          BackOrderShipment : ShipmentObj;
          Difference : REAL;
          Destination : BaseObj;
          Route : BaseQObj;

     BEGIN
```

```
OUTPUT("IN FillOrder - ", Name);

          Receiver := Shipment.Destination;
          ReceiverItem := Shipment.Item;

{determine OrderQty}

          OrderQty := ReceiverItem.StockTo - (ReceiverItem.OnOrder +
                                              ReceiverItem.OnHand);

IF (OrderQty >= 1.0)

{find item in inventory}

           MyItem := ASK Inventory TO FindByName(ReceiverItem.Name ;
          IF MyItem = NILOBJ
                MyItem := CLONE(ReceiverItem);
                ASK MyItem TO SetOnHand(0.0);
                ASK MyItem TO SetOnOrder(0.0);
                ASK MyItem TO SetStockTo(ReceiverItem.StockTo - ReceiverItem.
                                                                  OnHand);
                ASK MyItem TO SetOrderAt(MyItem.StockTo * .75);
                ASK MyItem TO SetEmerOrderAt(MyItem.StockTo * .50);
                ASK Inventory TO Add(MyItem);
          END IF;

{make copies of Receiver Item}

          IF MyItem.StockTo < OrderQty
                ASK MyItem TO SetStockTo(OrderQty);
                ASK MyItem TO SetOrderAt(MyItem.StockTo * .75);
                ASK MyItem TO SetEmerOrderAt(MyItem.StockTo * .50);
          END IF;

          MyOnHand := ASK MyItem OnHand;

{Case I - There is sufficient on hand}

           IF MyOnHand >= OrderQty

{Create and Route Shipment}

               ASK Shipment.Item TO SetOnHand(OrderQty);
               ASK MyItem TO SubtractOnHand(OrderQty);
               ASK SELF TO RouteShipments(Shipment);
           ELSIF MyOnHand < 1.0

{Case II - There is none, place whole order on BackOrder}

               ASK SELF TO BackOrderStuff(Shipment);
               MyStockTo := ASK MyItem StockTo;
               IF MyStockTo = 0.0
                    ASK MyItem TO SetStockTo(OrderQty);
               END IF;
           ELSE
{Case III -  Send what there is and back order the remainder}

                NEW(BackOrderShipment);
                Destination := ASK Shipment Destination;
                ASK BackOrderShipment TO SetDestination(Destination);
                ASK BackOrderShipment TO SetRDD(Shipment.RDD);
                Route := ASK Shipment Route;
                ASK BackOrderShipment TO SetRoute(Route);
                ASK BackOrderShipment TO SetItem(Shipment.Item);
               ASK Shipment.Item TO SetOnHand(MyOnHand);
               ASK BackOrderShipment.Item TO SetOnHand(0.0);
               ASK BackOrderShipment.Item TO AddOnOrder(MyOnHand);
               ASK SELF TO BackOrderStuff(BackOrderShipment);
               ASK MyItem TO SubtractOnHand(ASK Shipment.Item OnHand);
               ASK SELF TO RouteShipments(Shipment);

          END IF;

{WriteLine("Filling Order for " + REALTOSTR(OrderQty) + " " + ReceiverItem.Name

ELSE
        DISPOSE(Shipment);
END IF;

     END METHOD;

{----------------------------------------------------------------------------}
     ASK METHOD ReceiveStuff(INOUT Cargo : ShipmentQObj);
{----------------------------------------------------------------------------}

{method receives shipments destined for the base Inventory}

     VAR
          Shipment : ShipmentObj;
          ItemName : STRING;
          InventoryItem : CommodityObj;
          ShippedQty : REAL;
          i, numItems : INTEGER;
     BEGIN

OUTPUT("IN ReceiveStuff - ", Name);

{get first cargo item, repeat until no more cargo}

        numItems := ASK Cargo numberIn;

{find out what the cargo is and how much}

        FOR i := 1 TO numItems

                Shipment := ASK Cargo Remove;
                IF Shipment.Destination = SELF

                        ItemName := ASK Shipment.Item Name;
                        ShippedQty := ASK Shipment.Item OnHand;

                                                                {get the current
                                                                inventory object and add
                                                                 shipment}

                        InventoryItem := ASK Inventory TO FindByName(ItemName);
                        IF InventoryItem = NILOBJ
                                InventoryItem := CLONE(Shipment.Item);
```

```
                                        ASK Inventory TO Add(InventoryItem);
                                ELSE
                                        ASK InventoryItem TO AddOnHand(ShippedQty);
                                        ASK InventoryItem TO
                                                        SubtractOnOrder(ShippedQty);
                                END IF;

{
OUTPUT("Received Item - ",InventoryItem.Name," ",InventoryItem.OnHand," ",
InventoryItem.StockTo, " ",InventoryItem.OnOrder);
}
{WriteLine("Receiving Shipment at time:");
}
                                DISPOSE(Shipment);
                        ELSE
                                ASK Shipment.Route TO RemoveThis(Shipment.Route.First);
                                ASK SELF TO RouteShipments(Shipment);
                        END IF;
                END FOR;
                ASK SELF TO FillBackOrders;
{fill orders now that there is more stuff}
     END METHOD;

{--------------------------------------------------------------------------}
 ASK METHOD BackOrderStuff(INOUT BackOrderShipment : ShipmentObj);
{--------------------------------------------------------------------------}

{adds a Shipment to the backorder list}

     VAR

BEGIN

{
OUTPUT("IN BackOrderStuff - ", Name);
}

ASK BackOrders TO Add(BackOrderShipment);

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD FillBackOrders;
{--------------------------------------------------------------------------}

{method called after receiving shipment, checks to see if backorders can be
filled,  fills those that can}

VAR

CurrentBackOrder : ShipmentObj;
OrderName : STRING;
BackOrderItem : CommodityObj;
MyItem : CommodityObj;
MyOnHand : REAL;
i, numItems : INTEGER;

BEGIN
{
```

```
OUTPUT("IN FillBackOrders - ", Name);
}

{Go Thru Each order}

            numItems := ASK BackOrders numberIn;
            FOR i := 1 TO numItems
                   CurrentBackOrder := ASK BackOrders TO Remove;
                   BackOrderItem := ASK CurrentBackOrder Item;
                   MyItem := ASK Inventory TO FindByName(BackOrderItem.Name);
                   MyOnHand := ASK MyItem OnHand;

{Check if the commodity now is onhand}

                   IF MyOnHand >= 1.0

{if it is, ASK SELF TO FillOrder with BackOrder Fields}

                        ASK SELF TO FillOrder(CurrentBackOrder);
                  ELSE
{or put it back in the Queue}

                        ASK BackOrders TO Add(CurrentBackOrder);
                  END IF;
            END FOR;
END METHOD;


{----------------------------------------------------------------------------}
     ASK METHOD RouteShipments(INOUT Shipment : ShipmentObj);
{----------------------------------------------------------------------------}

{method send a Shipment to the proper port for transportation}

VAR

CargoPriority : INTEGER;
Receiver : BaseObj;
Commodity : CommodityObj;

BEGIN

Receiver := ASK Shipment.Route First;
CargoPriority := ASK Shipment.Item Priority;

OUTPUT("IN Route Shipments - ", Name, " to " , Receiver.Name );
OUTPUT("Cargo Priority = ", Shipment.Item.Priority);

IF (CargoPriority < 4)
        IF (Receiver.HasAirPort AND HasAirPort)
                ASK AirPort TO SortCargo(Shipment);

        ELSIF (Receiver.HasRail AND HasRail) AND (ASK RailYard.Network TO
                                                        Includes (Receiver));
                ASK RailYard TO SortCargo(Shipment);

        ELSIF (Receiver.HasTruckStop AND HasTruckStop) AND (ASK
         TruckStop.Network TO Includes(Receiver));
                ASK TruckStop TO SortCargo(Shipment);
        ELSIF (Receiver.HasSeaPort AND HasSeaPort)
                ASK SeaPort TO SortCargo(Shipment);

        ELSE
                        OUTPUT("NO ROUTE CHOSEN - BASEOBJ/ROUTESHIPMENTS");
                        HALT;
                                                        {DO SOMETHING HERE}
        END IF;

ELSIF (CargoPriority > 3) AND (CargoPriority < 7)
        IF (Receiver.HasRail AND HasRail) AND (ASK RailYard.Network TO
                                                        Includes (Receiver));
                ASK RailYard TO SortCargo(Shipment);

        ELSIF (Receiver.HasTruckStop AND HasTruckStop) AND (ASK
         TruckStop.Network TO Includes(Receiver));
                ASK TruckStop TO SortCargo(Shipment);

        ELSIF (Receiver.HasSeaPort AND HasSeaPort)
                ASK SeaPort TO SortCargo(Shipment);

        ELSIF (Receiver.HasAirPort AND HasAirPort)
                ASK AirPort TO SortCargo(Shipment);

        ELSE
                OUTPUT("No chosen routing");
                HALT;
                                                        {DO SOMETHING HERE}
        END IF;

ELSIF (CargoPriority > 6) AND (CargoPriority < 10)
        IF (Receiver.HasTruckStop AND HasTruckStop) AND (ASK TruckStop.Network
         TO Includes(Receiver));
                ASK TruckStop TO SortCargo(Shipment);

        ELSIF (Receiver.HasSeaPort AND HasSeaPort)
                ASK SeaPort TO SortCargo(Shipment);

        ELSIF (Receiver.HasRail AND HasRail) AND (ASK RailYard.Network TO
                                                Includes (Receiver));
                ASK RailYard TO SortCargo(Shipment);

        ELSIF (Receiver.HasAirPort AND HasAirPort)
                ASK AirPort TO SortCargo(Shipment);

        ELSE
                OUTPUT("No chosen routing");
                HALT;
                                                        {DO SOMETHING HERE}
        END IF;

ELSIF (CargoPriority < 9)
        IF (Receiver.HasSeaPort AND HasSeaPort)
                ASK SeaPort TO SortCargo(Shipment);
        ELSIF (Receiver.HasTruckStop AND HasTruckStop) AND (ASK
         TruckStop.Network TO Includes(Receiver));
                ASK TruckStop TO SortCargo(Shipment);

        ELSIF (Receiver.HasRail AND HasRail) AND (ASK RailYard.Network TO
                                                Includes (Receiver));
                ASK RailYard TO SortCargo(Shipment);

        ELSIF (Receiver.HasAirPort AND HasAirPort)
                ASK AirPort TO SortCargo(Shipment);
        ELSE
                OUTPUT("No chosen routing");
                HALT;
                                                                {DO SOMETHING HERE}
        END IF;
END IF;
END METHOD;

{--------------------------------------------------------------------------}
     TELL METHOD CheckInventory;
{--------------------------------------------------------------------------}

{method is 24.0 simtime check of inventory}

VAR

CurrentItem : CommodityObj;
i, numItems : INTEGER;
integer : INTEGER;
BEGIN
{
ASK SELF TO DumpFields;
}
LOOP

WAIT DURATION 24.0

{
OUTPUT("IN CheckInventory - ", Name);
}
        numItems := ASK Inventory numberIn;
        FOR i := 1 TO numItems
                CurrentItem := ASK Inventory TO Remove;
                ASK Inventory TO Add(CurrentItem);

{Set Higher Priority if necessary.}

                IF CurrentItem.OnHand <= CurrentItem.EmerOrderAt
                        ASK CurrentItem TO
                                         SetPriority(CurrentItem.EmerPriority);
                ELSE
                        ASK CurrentItem TO
                                        SetPriority(CurrentItem.NormalPriority);
                END IF;

{Order If Necessary}

                IF CurrentItem.OnHand + CurrentItem.OnOrder <=
                                                        CurrentItem.OrderAt
                        ASK SELF TO OrderStuff(CurrentItem);
                END IF;
        END FOR;

{
WriteLine("CheckingInventory at time:");
ASK SELF TO DumpFields;
```

```
}

END WAIT;
END LOOP;

END METHOD;


{----------------------------------------------------------------------------}
     ASK METHOD DumpFields;
{----------------------------------------------------------------------------}

{dumps Inventory to sim.out file}

VAR

i, numItems : INTEGER;
Transporter : TransporterObj;
Commodity : CommodityObj;
BEGIN
WriteLine(" ");
WriteLine("========================="+REALTOSTR(SimTime)+"====================
WriteLine(" ");

WriteLine("Base Name = "+ Name);
WriteLine(" ");


{
IF HasAirPort
        WriteLine("HasAirport:");
        WriteLine("              BerthsQ:");
        numItems := ASK AirPort.BerthsQ numberIn;
        FOR i := 1 TO numItems;
                Transporter := ASK AirPort.BerthsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK AirPort.BerthsQ TO Add(Transporter);
        END FOR;
        WriteLine("              ArrivalsQ:");
        numItems := ASK AirPort.ArrivalsQ numberIn;
        FOR i := 1 TO numItems;
                Transporter := ASK AirPort.ArrivalsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK AirPort.ArrivalsQ TO Add(Transporter);
        END FOR;
        WriteLine(" ");

END IF;

IF HasSeaPort
        WriteLine("HasSeaport:");
        WriteLine("              BerthsQ:");
        numItems := ASK SeaPort.BerthsQ numberIn;
        FOR i := 1 TO numItems;
                Transporter := ASK SeaPort.BerthsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK SeaPort.BerthsQ TO Add(Transporter);
        END FOR;
        WriteLine("              ArrivalsQ:");
        numItems := ASK SeaPort.ArrivalsQ numberIn;
        FOR i := 1 TO numItems;
                Transporter := ASK SeaPort.ArrivalsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK SeaPort.ArrivalsQ TO Add(Transporter);
        END FOR;
        WriteLine(" ");
END IF;


IF HasRail
        WriteLine("HasRail:");
        WriteLine("           BerthsQ:");
        numItems := ASK RailYard.BerthsQ numberIn;
        FOR i := 1 TO numItems;
                Transporter := ASK RailYard.BerthsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK RailYard.BerthsQ TO Add(Transporter);
        END FOR;
        WriteLine("           ArrivalsQ:");
        numItems := ASK RailYard.ArrivalsQ numberIn;
        FOR i := 1 TO numItems;
                Transporter := ASK RailYard.ArrivalsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK RailYard.ArrivalsQ TO Add(Transporter);
        END FOR;
        WriteLine(" ");
END IF;

IF HasTruckStop
        WriteLine("HasTruckStop:");
        WriteLine("           BerthsQ:");
        numItems := ASK TruckStop.BerthsQ numberIn;
        FOR i := 1 TO numItems;
                Transporter := ASK TruckStop.BerthsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK TruckStop.BerthsQ TO Add(Transporter);
        END FOR;
        WriteLine("           ArrivalsQ:");
        numItems := ASK TruckStop.ArrivalsQ numberIn;

        FOR i := 1 TO numItems;
                Transporter := ASK TruckStop.ArrivalsQ TO Remove;
                WriteLine(" "+Transporter.Name+INTTOSTR(Transporter.VehicleID));
                ASK TruckStop.ArrivalsQ TO Add(Transporter);
        END FOR;
        WriteLine(" ");
END IF;
}

numItems := ASK Inventory numberIn;

WriteLine("Inventory:  "+ INTTOSTR(numItems));

FOR i := 1 TO numItems
        Commodity := ASK Inventory TO Remove;
        WriteLine(INTTOSTR(i)+". "+ Commodity.Name+": OnHand - "
      +REALTOSTR(Commodity.OnHand)+" OnOrder - "+REALTOSTR(Commodity.OnOrder)+"
        ASK Inventory TO Add(Commodity);
END FOR;
```

```
WriteLine("================================================================");
WriteLine(" ");

END METHOD;

{-------------------------------------------------------------------------}
     ASK METHOD Display;
{-------------------------------------------------------------------------}

{displays base to screen}

CONST

format = "          *********** ********** ********** ********** ***********";

format2 = "***. *************** ********* ********* ********* *********";
title = "====================== ***************> at time ******<  =============

VAR

j, i, numItems : INTEGER;
Transporter : TransporterObj;
Commodity : CommodityObj;
string, answer : STRING;
CHR : CHAR;

BEGIN



ClearScreen;
j := 0;
SOUTPUT(" ",j);
string := SPRINT(Name, TRUNC(SimTime)) WITH title;
SOUTPUT(string,j);
SOUTPUT(" ",j);
OutputPosition(Position, j);
SOUTPUT(" ", j);
OUTPUT("Group:  ", Group);
OUTPUT("Subgroup:  ", SubGroup);

IF HasAirPort
        SOUTPUT(" ", j);
        SOUTPUT("Airport:                 Max Capacity: "
                 + INTTOSTR(AirPort.MaxCapacity) + "    Max Vehicle Size: "
                 + REALTOSTR(AirPort.MaxSize),j);
        OUTPUT("            Arrivals:  ", AirPort.ArrivalsQ.numberIn);
        OUTPUT("            Ramp:      ", AirPort.BerthsQ.numberIn);
        OUTPUT("            Parked:    ", AirPort.ParkedQ.numberIn);
        j := j + 3;

END IF;

IF HasSeaPort
        SOUTPUT(" ",j);
        SOUTPUT("Seaport:                 Max Capacity: "
                 + INTTOSTR(SeaPort.MaxCapacity)+ "    Max Vehicle Size: "
                 + REALTOSTR(SeaPort.MaxSize),j);
        OUTPUT("            Arrivals:  ", SeaPort.ArrivalsQ.numberIn);
        OUTPUT("            Berths:    ", SeaPort.BerthsQ.numberIn);
        OUTPUT("          Anchorage: ", SeaPort.ParkedQ.numberIn);
        j := j + 3;

END IF;


IF HasRail
        SOUTPUT(" ",j);
        SOUTPUT("Railyard:              Max Capacity: "+
                INTTOSTR(RailYard.MaxCapacity)+ "     Max Vehicle Size: "+
                                                REALTOSTR(RailYard.MaxSize),j);
        OUTPUT("          Arrivals: ", RailYard.ArrivalsQ.numberIn);
        OUTPUT("          Station:  ", RailYard.BerthsQ.numberIn);
        OUTPUT("          Yard:     ", RailYard.ParkedQ.numberIn);
        j := j + 3;
END IF;

IF HasTruckStop
        SOUTPUT(" ",j);
        SOUTPUT("Truck Stop:            Max Capacity: "
                 + INTTOSTR(TruckStop.MaxCapacity)+ "     Max Vehicle Size: "
                 + REALTOSTR(TruckStop.MaxSize), j);
        OUTPUT("          Arrivals:  ", TruckStop.ArrivalsQ.numberIn);
        OUTPUT("          Terminal:  ", TruckStop.BerthsQ.numberIn);
        OUTPUT("          Parked:    ", TruckStop.ParkedQ.numberIn);
        j := j + 3;
END IF;

SOUTPUT(" ",j);
numItems := ASK Inventory numberIn;
SOUTPUT("Items in Inventory:  "+INTTOSTR( numItems),j);
SOUTPUT("                           ON HAND  STOCK TO  ORDER AT  ON ORDER",j);

FOR i := 1 TO numItems
        Commodity := ASK Inventory TO Remove;
        ASK Inventory TO Add(Commodity);
string := SPRINT(i, Commodity.Name, TRUNC(Commodity.OnHand),
         TRUNC(Commodity.StockTo), TRUNC(Commodity.OrderAt),
         TRUNC(Commodity.OnOrder)) WITH format2;
SOUTPUT(string,j);
END FOR;

SOUTPUT("=================================================================
SOUTPUT(" ",j);


END METHOD;

{-----------------------------------------------------------------------}
     ASK METHOD Modify(INOUT builder : BuilderObj);
{-----------------------------------------------------------------------}

{allows interactive modification of base in execution or scenario construction}

VAR

base : BaseObj;
string, answer, answer2: STRING;
NewPort, port : PortObj;
commodity : CommodityObj;
```

```
integer : INTEGER;
real : REAL;
j, i, numItems : INTEGER;
CHR, CHR2 : CHAR;
position : PositionRecType;
TransporterQ, BigTransporterQ : TransporterQObj;
Transporter, NewTransporter : TransporterObj;

BEGIN


        LOOP
                ASK SELF TO Display;
                OUTPUT("      MODIFY?  (P)orts, (I)nventory,  (S)tatus, (R)eturn
                CHR := ReadKey();
                IF (CHR = "P") OR (CHR = "p")
                        LOOP

                        ClearScreen;
                        OUTPUT(" ");
                        OUTPUT("      ",SELF.Name, " has the following ports:");
                        OUTPUT(" ");
                        OUTPUT("           Airport:     ", SELF.HasAirPort);
                        OUTPUT("           Seaport:     ", SELF.HasSeaPort);
                        OUTPUT("           Railyard:    ", SELF.HasRail);
                        OUTPUT("           Truck Stop:  ", SELF.HasTruckStop);
                        OUTPUT(" ");
                        OUTPUT("      MODIFY?   (A)irport, (S)eaport, Rail(y)ard,
                        CHR2 := ReadKey();
                        IF (CHR2 = "A") OR (CHR2 = "a")
                                port := ASK SELF AirPort;
                        ELSIF (CHR2 = "S") OR (CHR2 = "s")
                                port := ASK SELF SeaPort;
                        ELSIF (CHR2 = "Y") OR (CHR2 = "y")
                                port := ASK SELF RailYard;
                        ELSIF (CHR2 = "T") OR (CHR2 = "t")
                                port := ASK SELF TruckStop;
                        ELSIF (CHR2 = "R") OR (CHR2 = "r");
                                EXIT;
                        END IF;
                        OUTPUT("MODIFY? (E)xistence, Max (C)apacity, Max Vehicle
                        CHR := ReadKey();
                        OUTPUT(" ");
                        IF (CHR = "E") OR (CHR = "e")
                                IF (CHR2 = "A") OR (CHR2 = "a")
                                        IF SELF.HasAirPort
                                        ASK SELF TO SetHasAirPort("FALSE");
                                        ELSE
                                        ASK SELF TO SetHasAirPort("TRUE");
                                        END IF;
                                ELSIF (CHR2 = "S") OR (CHR2 = "s")
                                        IF SELF.HasSeaPort
                                        ASK SELF TO SetHasSeaPort("FALSE");
                                        ELSE
                                        ASK SELF TO SetHasSeaPort("TRUE");
                                        END IF;
                                ELSIF (CHR2 = "Y") OR (CHR2 = "y")
                                        IF SELF.HasRail
                                        ASK SELF TO SetHasRail("FALSE");
                                        ELSE
```

```
                    ASK SELF TO SetHasRail("TRUE");
                    END IF;
                ELSIF (CHR2 = "T") OR (CHR2 = "t")
                    IF SELF.HasTruckStop
                    ASK SELF TO SetHasTruckStop("FALSE");
                    ELSE
                    ASK SELF TO SetHasTruckStop("TRUE");
                    END IF;
                END IF;


            ELSIF (CHR = "C") OR (CHR = "c")
                OUTPUT("     Enter New Max Capacity and hit <ENT
                INPUT(integer);
                ASK port TO SetMaxCapacity(integer);
            ELSIF (CHR = "S") OR (CHR = "s")
                OUTPUT("     Enter New Max Vehicle Size (Real)."
                OUTPUT("     Units:  Air   - Square foot area");
                OUTPUT("             Ship  - Overall length (fee
                OUTPUT("             Rail  - Length in cars.");
                OUTPUT("             Truck - Vehicles in Convoy.
                INPUT(real);
                ASK port TO SetMaxSize(real);

            ELSIF (CHR = "N") OR (CHR = "n")
            LOOP
                ClearScreen;
                OUTPUT(" ");
                OUTPUT("     Transportation network includes the
                OUTPUT(" ");
                numItems := ASK port.Network numberIn;
                IF numItems <> 0
                FOR i := 1 TO numItems
                    base := ASK port.Network TO Remove;
                    ASK port.Network TO Add(base);
                    OUTPUT(base.Name);
                END FOR;
                ELSE
                    OUTPUT("     NONE");
                END IF;
                OUTPUT(" ");
                OUTPUT("     COMMAND:  (A)dd, (S)ubtract, (R)etu
                CHR2 := ReadKey();

                IF (CHR2 = "A") OR (CHR2 = "a")
                    OUTPUT("Base or Unit Name?");
                    INPUT(answer);
                    base := ASK builder.BaseQ TO
                             FindByName(answer);
                    IF base <> NILOBJ
                            ASK port.Network TO Add(base);
                    END IF;
                ELSIF (CHR2 = "S") OR (CHR2 = "s")
                    OUTPUT("Base or Unit Name?");
                    INPUT(answer);
                    base := ASK builder.BaseQ TO
                             FindByName(answer);
                    IF base <> NILOBJ
                            ASK port.Network TO
                                    RemoveThis(base);
                    END IF;
                ELSIF (CHR2 = "R") OR (CHR2 = "r")
                    EXIT;
                END IF;
            END LOOP;
            ELSIF (CHR = "T") OR (CHR = "t")
                OUTPUT("     (A)dd or (D)elete");
                CHR := ReadKey();
                IF (CHR = "A") OR (CHR = "a")
                    ClearScreen;
                    j := 0;
                    IF Builder <> NILOBJ
                        TransporterQ := Builder.TransporterQ;
                        BigTransporterQ :=
                                 Builder.BigTransporterQ;
                    ELSIF ScenarioEditor <> NILOBJ
                        TransporterQ :=
                                 ScenarioEditor.TransporterQ;
                        BigTransporterQ :=
                                 ScenarioEditor.BigTransporterQ;
                    END IF;
                    IF TransporterQ <> NILOBJ
                        ClearScreen;
                        j := 0;
                        ASK TransporterQ TO Display(j);
                        OUTPUT("     Input transporter name.");
                        INPUT(string);
                        Transporter := ASK TransporterQ TO
                                 FindByName(string);
                        IF Transporter <> NILOBJ
                            OUTPUT("     How many?");
                            INPUT(integer);

                            FOR i := 1 TO integer
                                NewTransporter :=
                                 CLONE(Transporter);
                                ASK Transporter TO
                        SetVehicleID(Transporter.VehicleID + 1);
                                ASK NewTransporter TO
                            SetVehicleID(Transporter.VehicleID);
                                ASK NewTransporter TO
                                 SetLocation(SELF);
                                ASK NewTransporter TO
                                 SetPosition(Position);
                                ASK NewTransporter TO
                                 SetPort(SELF);
                                ASK BigTransporterQ TO
                                 Add(NewTransporter);
                                CASE ASK NewTransporter Class
                                WHEN Aircraft :
                                        NewPort := AirPort;
                                WHEN Ship :
                                        NewPort := SeaPort;
                                WHEN Rail :
                                        NewPort := RailYard;
                                WHEN Truck :
                                        NewPort := TruckStop;
                                END CASE;
                                ASK NewPort.ParkedQ TO
                                 Add(NewTransporter);
```

```
                                    ASK TransporterManager TO
                        ReceiveAvailableTransporter(NewTransporter);
                                END FOR;
                            END IF;
                        END IF;
                    ELSIF (CHR = "D") OR (CHR = "d")
                        ClearScreen;
                        j := 0;
                        ASK port.ParkedQ TO Display(j);
                        OUTPUT("     Input Name");
                        INPUT(string);
                        OUTPUT("     Input ID");
                        INPUT(integer);
                        numItems := ASK port.ParkedQ numberIn;
                        FOR i := 1 TO numItems
                            Transporter := ASK port.ParkedQ TO
                                    Remove;
                            ASK port.ParkedQ TO Add(Transporter);
                            IF Transporter.VehicleID = integer;
                                    ASK port.ParkedQ TO
                                            RemoveThis(Transporter);
                                    OUTPUT(Transporter.Name,
                                             Transporter.VehicleID,
                                             " deleted");
                                    ASK Transporter TO CleanUp;
                            END IF;
                        END FOR;
                    END IF;
                ELSIF (CHR = "R") OR (CHR = "r")
                    EXIT
                END IF;

                END LOOP;


        ELSIF (CHR = "I") OR (CHR = "i")
            OUTPUT("     (A)dd Commodity, (E)dit Commodity?");
            CHR := ReadKey();
            IF (CHR = "A") OR (CHR = "a")
                InputCommodities(builder);
            ELSIF (CHR = "E") OR (CHR = "e")
                OUTPUT("     Enter Commodity Name then hit <ENTER>");
                INPUT(string);
                commodity := ASK SELF.Inventory TO FindByName(string);
                OUTPUT(" ");
                IF commodity <> NILOBJ
                        LOOP

                        OUTPUT("     MODIFY? (O)n Hand, (S)tocking Objec
                        CHR := ReadKey();


                        IF (CHR = "O") OR (CHR = "o")
                                OUTPUT("     Input new amount On Hand.")
                                INPUT(real);
                                ASK commodity TO SetOnHand(real);
                        ELSIF (CHR = "S") OR (CHR = "s")
                                OUTPUT("     Input new Stocking Objectiv
                                INPUT(real);
                                ASK commodity TO SetStockTo(real);
```

```
                                        ELSIF (CHR = "p") OR (CHR = "P")
                                                OUTPUT("     Input new Order Point.");
                                                INPUT(real);
                                                ASK commodity TO SetOrderAt(real);
                                        ELSIF (CHR = "R") OR (CHR = "r")
                                                EXIT;

                                        END IF;
                                        END LOOP;
                                END IF;
                        END IF;

                ELSIF (CHR = "S") OR (CHR = "s")
                        LOOP

                        OUTPUT("     MODIFY? (P)osition, (G)roup, (S)ubGroup, (R
                        CHR := ReadKey();
                        IF (CHR = "P") OR (CHR = "p")
                                ClearScreen;
                                j := 0;
                                OutputPosition(Position, j);
                                position := InputPosition();
                                ASK SELF TO SetPosition(position);
                                OutputPosition(Position, j);
                        ELSIF (CHR = "G") OR (CHR = "g")
                                InputGroup;
                        ELSIF (CHR = "S") OR (CHR = "s")
                                InputSubGroup;
                        ELSIF (CHR = "R") OR (CHR = "r")
                                EXIT;
                        END IF;

                        END LOOP;

                ELSIF (CHR = "T") OR (CHR = "t")


                ELSIF (CHR = "R") OR (CHR = "r")
                        EXIT;
                END IF;
        END LOOP;

     END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD InputCommodities(INOUT builder : BuilderObj);
{----------------------------------------------------------------------------}

{inputs commodities during interactive construction of the Base}

VAR

commodity, newcommodity, commodity2 : CommodityObj;
string : STRING;
j, integer : INTEGER;
real : REAL;
CHR : CHAR;

BEGIN
LOOP
```

```
        ClearScreen;
        j := 0;
        ASK builder.CommodityQ TO Display(j);
        SOUTPUT("     Input Commodity Name",j);
        INPUT(string);
        commodity2 := ASK Inventory TO FindByName(string);
        IF commodity2 <> NILOBJ
                OUTPUT(string + " already in inventory.");
        ELSE
                commodity := ASK builder.CommodityQ TO FindByName(string);
                IF commodity <> NILOBJ
                        newcommodity := CLONE(commodity);
                        SOUTPUT("     How much do you want on hand?",j);
                        INPUT(real);
                        ASK newcommodity TO SetOnHand(real);
                        SOUTPUT("     How much do you the base or unit to"
                                + " stock?",j);
                        INPUT(real);
                        ASK newcommodity TO SetStockTo(real);
                        SOUTPUT("     At what level do you want the base or"
                                +" unit to Order more of the commodity?",j);
                        INPUT(real);
                        ASK newcommodity TO SetOrderAt(real);
                        ASK newcommodity TO SetEmerOrderAt(.25 *
                                newcommodity.StockTo);
                        ASK Inventory TO Add(newcommodity);
                END IF;
        END IF;
        SOUTPUT("     Add another commodity (Y).",j);
        CHR := ReadKey();
        IF (CHR = "n") OR (CHR = "N")
                EXIT;
        END IF;
END LOOP;
END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD ObjInit;
{---------------------------------------------------------------------------}

VAR

BEGIN

INHERITED ObjInit;

NEW(BackOrders);
NEW(AirPort);
ASK AirPort TO SetOwner(SELF);
NEW(SeaPort);
ASK SeaPort TO SetOwner(SELF);
NEW(RailYard);
ASK RailYard TO SetOwner(SELF);
NEW(TruckStop);
ASK TruckStop TO SetOwner(SELF);

END METHOD;


{---------------------------------------------------------------------------}
```

```
     ASK METHOD DisposePorts;
{---------------------------------------------------------------------------}
VAR


BEGIN
IF BackOrders <> NILOBJ
        DISPOSE(BackOrders);
END IF;
IF AirPort <> NILOBJ
        DISPOSE(AirPort);
END IF;
IF SeaPort <> NILOBJ
        DISPOSE(SeaPort);
END IF;
IF RailYard <> NILOBJ
        DISPOSE(RailYard);
END IF;
IF TruckStop <> NILOBJ
        DISPOSE(TruckStop);
END IF;
END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD ObjTerminate;
{---------------------------------------------------------------------------}
VAR


BEGIN

IF BackOrders <> NILOBJ
        DISPOSE(BackOrders);
END IF;
IF AirPort <> NILOBJ
        DISPOSE(AirPort);
END IF;
IF SeaPort <> NILOBJ
        DISPOSE(SeaPort);
END IF;
IF RailYard <> NILOBJ
        DISPOSE(RailYard);
END IF;
IF TruckStop <> NILOBJ
        DISPOSE(TruckStop);
END IF;
INHERITED ObjTerminate;

END METHOD;

END OBJECT;

END MODULE.
```

```
DEFINITION MODULE Builder;

{Scenario Builder}

{Import statements}

FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM IOMod IMPORT StreamObj;
{FROM RecIOHandle IMPORT RecIOHandleObj}
{FROM Distant IMPORT PositionRecType}
FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM Port IMPORT PortObj;
FROM Trnsprt IMPORT TransporterObj,
                    TransporterQObj,
                    ALL TransporterClassType;
FROM CommodQ IMPORT CommodityQObj,
                    CommodityObj;

FROM Supply IMPORT SupplyObj;
FROM Unit IMPORT UnitObj,
                 UnitQObj;

{Type Declarations}

TYPE

{===========================================================================}
BuilderObj = OBJECT
{===========================================================================}

{Builder object builds objects from datafiles to use in executing scenarios}

{FIELDS}
BaseQ : BaseQObj;
OnlyBaseQ : BaseQObj;
UnitQ : UnitQObj;
CurrentBase : BaseObj;
CommodityQ : CommodityQObj;
TransporterQ : TransporterQObj;
BigTransporterQ : TransporterQObj;
PrepoTransporterQ : TransporterQObj;
PrepoQ : MyQueueObj;

Supply : SupplyObj;

{METHODS}

ASK METHOD BuildBase(IN FileName : STRING) : BaseObj;
ASK METHOD BuildTransporter(IN FileName : STRING) : TransporterObj;
ASK METHOD BuildUnit(IN FileName : STRING) : UnitObj;
ASK METHOD BuildSupply;
ASK METHOD BuildPrepo(IN FileName : STRING);
ASK METHOD BuildScenario(IN FileName : STRING);
ASK METHOD BuildScenarioTransporters(IN FileName : STRING);
ASK METHOD BuildScenarioCommodities(IN FileName : STRING; INOUT commodityQ : Com
ASK METHOD BuildScenarioBases(IN FileName : STRING);
ASK METHOD BuildScenarioUnits(IN FileName : STRING);
ASK METHOD BuildScenarioPrepos(IN FileName : STRING);

ASK METHOD LinkUnits(IN FileName : STRING);
ASK METHOD LinkRailRoads(IN FileName : STRING);
ASK METHOD LinkRoads(IN FileName : STRING);
ASK METHOD ObjInit;
ASK METHOD ObjTerminate;
END OBJECT;

VAR

Builder : BuilderObj;

END MODULE.
```

```
IMPLEMENTATION MODULE Builder;

{Comments}

{Import statements}

FROM Trash IMPORT GarbageDisposal,
                  TrashCan;

FROM Distant IMPORT PositionRecType;
FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM IOMod IMPORT StreamObj,
                  ALL FileUseType;
FROM Base IMPORT BaseObj,
                 BaseQObj,
                 ALL BaseGroupType,
                 ALL BaseSubGroupType;
FROM Port IMPORT PortObj;
FROM Trnsprt IMPORT TransporterObj,
                    TransporterQObj,
                    ALL TransporterClassType;
FROM CommodQ IMPORT CommodityQObj,
                    CommodityObj;
FROM TManage IMPORT TransporterManager;
FROM Shpmnt IMPORT ShipmentObj;

FROM Supply IMPORT Supply;
FROM Unit IMPORT UnitObj,
                 UnitQObj;
FROM SimMod IMPORT InterruptAll,
                   Interrupt,
                   NumActivities;
FROM LogMan IMPORT LogisticsManager;

{Definitions}

{===========================================================================}
OBJECT BuilderObj;
{===========================================================================}

     {METHODS}
{---------------------------------------------------------------------------}
     ASK METHOD BuildBase(IN FileName : STRING) : BaseObj;
{---------------------------------------------------------------------------}

{builds base from base file}

VAR

File : StreamObj;
Base : BaseObj;
Port : PortObj;
string : STRING;
integer : INTEGER;
position : PositionRecType;
Transporter : TransporterObj;
NewTransporter : TransporterObj;
NewPort : PortObj;
n, m, numtransporters, numitems : INTEGER;
```

```
Commodity, NewCommodity : CommodityObj;

BEGIN

NEW(File);
NEW(Base);
{
OUTPUT("Creating Base - Builder");
}
NEW(position);
ASK File TO Open(FileName, Input);

{read base name}

ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK Base TO SetName(string);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

{read group data}

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Base TO SetGroup(string);
ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Base TO SetSubGroup(string);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

{read position data}

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
position.LatDeg := integer;
ASK File TO ReadInt(integer);
position.LatMin := integer;
ASK File TO ReadString(string);
position.LatDir := string;
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
position.LongDeg := integer;
ASK File TO ReadInt(integer);
position.LongMin := integer;
ASK File TO ReadString(string);
position.LongDir := string;
ASK File TO ReadLine(string);

ASK Base TO SetPosition(position);
DISPOSE(position);
ASK File TO ReadLine(string);

{read port data}
```

```
ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Base TO SetHasAirPort(string);
        ASK Base.AirPort TO SetOwner(Base);
        ASK Base.AirPort TO SetClass("Aircraft");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Base.AirPort TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Base.AirPort TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Base TO SetHasSeaPort(string);
        ASK Base.SeaPort TO SetOwner(Base);
        ASK Base.SeaPort TO SetClass("Ship");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Base.SeaPort TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Base.SeaPort TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Base TO SetHasRail(string);
        ASK Base.RailYard TO SetOwner(Base);
        ASK Base.RailYard TO SetClass("Rail");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Base.RailYard TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Base.RailYard TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Base TO SetHasTruckStop(string);
        ASK Base.TruckStop TO SetOwner(Base);
        ASK Base.TruckStop TO SetClass("Truck");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Base.TruckStop TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Base.TruckStop TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

{read transporter data}

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numtransporters := integer;
ASK File TO ReadLine(string);
```

```
FOR m := 1 TO numtransporters
        ASK File TO ReadLine(string);
        ASK File TO ReadString(string);
        Transporter := ASK TransporterQ TO FindByName(string);

        ASK File TO ReadInt(integer);
        FOR n := 1 TO integer
                NewTransporter := CLONE(Transporter);
                ASK Transporter TO SetVehicleID(Transporter.VehicleID + 1);
                ASK NewTransporter TO SetVehicleID(Transporter.VehicleID);
                ASK NewTransporter TO SetLocation(Base);
                ASK NewTransporter TO SetDestination(Base);
                ASK NewTransporter TO SetPosition(Base.Position);
                ASK NewTransporter TO SetPort(Base);
                ASK BigTransporterQ TO Add(NewTransporter);


                CASE ASK NewTransporter Class
                        WHEN Aircraft :
                                NewPort := Base.AirPort;

                        WHEN Ship :
                                NewPort := Base.SeaPort;

                        WHEN Rail :
                                NewPort := Base.RailYard;

                        WHEN Truck :
                                NewPort := Base.TruckStop;
                END CASE;
                ASK NewPort.ParkedQ TO Add(NewTransporter);
                ASK TransporterManager TO
                                    ReceiveAvailableTransporter(NewTransporter);
{
                IF (NewPort.BerthsQ.numberIn) < (NewPort.MaxCapacity)
                        ASK NewPort.BerthsQ TO Add(NewTransporter);
                        ASK TransporterManager TO
                                    ReceiveAvailableTransporter(NewTransporter);
                ELSE
                        ASK NewPort.ArrivalsQ TO Add(NewTransporter);
                END IF;
}

        END FOR;
END FOR;

ASK File TO ReadLine(string);
ASK File TO ReadLine(string);

{read commodity data}

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numitems := integer;
ASK File TO ReadLine(string);

FOR m := 1 TO numitems

        ASK File TO ReadLine(string);
```

```
          ASK File TO ReadString(string);
          Commodity := ASK CommodityQ TO FindByName(string);
          ASK File TO ReadLine(string);

          IF Commodity <> NILOBJ
                  NewCommodity := CLONE(Commodity);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetOnHand(real);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetStockTo(real);
                  ASK File TO ReadLine(string);

                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetOrderAt(real);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetEmerOrderAt(real);
                  ASK File TO ReadLine(string);

                  ASK Base.Inventory TO Add(NewCommodity)
          ELSE
                  ASK File TO ReadLine(string);
                  ASK File TO ReadLine(string);
                  OUTPUT("Unlisted Commodity in Base file");
          END IF;

  END FOR;

  ASK File TO Close;
  DISPOSE(File);
  RETURN(Base);

  END METHOD;

  {-----------------------------------------------------------------------}
  ASK METHOD BuildTransporter(IN FileName : STRING) : TransporterObj;
  {-----------------------------------------------------------------------}

  {builds TransporterObj from datafile}


  CONST

  VAR

  File : StreamObj;
  string : STRING;
  integer : INTEGER;
  Position : PositionRecType;
  Transporter : TransporterObj;

  BEGIN

  NEW(File);
  NEW(Transporter);
```

```
ASK Transporter TO SetVehicleID(0);
ASK File TO Open(FileName, Input);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Transporter TO SetName(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Transporter TO SetClass(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Transporter TO SetSubClass(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetLength(real);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetWidth(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxSpeed(real);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxRange(real);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxCargoArea(real);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxCargoCube(real);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxCargoWeight(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxCargoLength(real);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxCargoWidth(real);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxCargoHeight(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxPax(real);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Transporter TO SetMaxGas(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Transporter TO SetOverSize(string);

ASK File TO Close;
DISPOSE(File);
RETURN(Transporter);

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD BuildScenarioCommodities(IN FileName : STRING; INOUT commodityQ : Com
{-----------------------------------------------------------------------}

{builds commodities from commodity master file}

VAR

File : StreamObj;
string : STRING;
integer : INTEGER;
Commodity : CommodityObj;
n, numItems : INTEGER;


BEGIN

NEW(File);
ASK File TO Open(FileName, Input);

ASK File TO ReadLine(string);
ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;
ASK File TO ReadLine(string);

FOR n := 1 TO numItems
        NEW(Commodity);
        ASK File TO ReadLine(string);
        ASK File TO ReadString(string);
        ASK File TO ReadString(string);
        ASK Commodity TO SetName(string);
        ASK File TO ReadString(string);
        ASK File TO ReadString(string);
        ASK Commodity TO SetClass(string);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Commodity TO SetProduceAt(real);
        ASK File TO ReadLine(string);

        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Commodity TO SetLength(real);
```

```
          ASK File TO ReadString(string);
          ASK File TO ReadReal(real);
          ASK Commodity TO SetWidth(real);
          ASK File TO ReadString(string);
          ASK File TO ReadReal(real);
          ASK Commodity TO SetHeight(real);
          ASK File TO ReadLine(string);

          ASK File TO ReadString(string);
          ASK File TO ReadReal(real);
          ASK Commodity TO SetWeight(real);
          ASK File TO ReadString(string);
          ASK File TO ReadInt(integer);
          ASK Commodity TO SetPriority(integer);
          ASK Commodity TO SetNormalPriority(integer);
          ASK File TO ReadString(string);
          ASK File TO ReadInt(integer);
          ASK Commodity TO SetEmerPriority(integer);
          ASK File TO ReadLine(string);

          ASK File TO ReadString(string);
          ASK File TO ReadString(string);
          ASK Commodity TO SetOverSize(string);
          ASK File TO ReadLine(string);

          ASK commodityQ TO Add(Commodity);
END FOR;
ASK File TO Close;
DISPOSE(File);

END METHOD;



{------------------------------------------------------------------------}
ASK METHOD BuildUnit(IN FileName : STRING) : UnitObj;
{------------------------------------------------------------------------}

{builds a UnitObj from a datafile}

VAR

File : StreamObj;
Unit : UnitObj;
Port : PortObj;
string : STRING;
integer : INTEGER;
position : PositionRecType;
Transporter : TransporterObj;
NewTransporter : TransporterObj;
NewPort : PortObj;
n, m, numtransporters, numitems : INTEGER;
Commodity, NewCommodity : CommodityObj;

BEGIN

NEW(File);
NEW(Unit);
{
```

```
OUTPUT("Creating Unit");
}
NEW(position);
ASK File TO Open(FileName, Input);

{read base name}

ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK Unit TO SetName(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Unit TO SetClass(string);
ASK File TO ReadLine(string);

ASK Unit TO SetGroup("UNIT");
ASK Unit TO SetSubGroup("NONE");
ASK File TO ReadLine(string);

{read position data}

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
position.LatDeg := integer;
ASK File TO ReadInt(integer);
position.LatMin := integer;
ASK File TO ReadString(string);
position.LatDir := string;
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
position.LongDeg := integer;
ASK File TO ReadInt(integer);
position.LongMin := integer;
ASK File TO ReadString(string);
position.LongDir := string;
ASK File TO ReadLine(string);

ASK Unit TO SetPosition(position);

DISPOSE(position);

{read Unit Data}

ASK File TO ReadLine(string);
ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Unit TO SetDelayUntil(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Unit TO SetInPlace(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadReal(real);
ASK Unit TO SetActiveAt(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Unit TO SetCombatIntensity(string);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

{read port data}

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Unit TO SetHasAirPort(string);
        ASK Unit.AirPort TO SetOwner(Unit);
        ASK Unit.AirPort TO SetClass("Aircraft");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Unit.AirPort TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Unit.AirPort TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Unit TO SetHasSeaPort(string);
        ASK Unit.SeaPort TO SetOwner(Unit);
        ASK Unit.SeaPort TO SetClass("Ship");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Unit.SeaPort TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Unit.SeaPort TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Unit TO SetHasRail(string);
        ASK Unit.RailYard TO SetOwner(Unit);
        ASK Unit.RailYard TO SetClass("Rail");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Unit.RailYard TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Unit.RailYard TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK Unit TO SetHasTruckStop(string);
        ASK Unit.TruckStop TO SetOwner(Unit);
        ASK Unit.TruckStop TO SetClass("Truck");
        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        ASK Unit.TruckStop TO SetMaxCapacity(integer);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK Unit.TruckStop TO SetMaxSize(real);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numtransporters := integer;
ASK File TO ReadLine(string);

FOR m := 1 TO numtransporters
        ASK File TO ReadLine(string);
        ASK File TO ReadString(string);
        Transporter := ASK TransporterQ TO FindByName(string);

        ASK File TO ReadInt(integer);
        FOR n := 1 TO integer
                NewTransporter := CLONE(Transporter);
                ASK Transporter TO SetVehicleID(Transporter.VehicleID + 1);
                ASK NewTransporter TO SetVehicleID(Transporter.VehicleID);
                ASK NewTransporter TO SetLocation(Unit);
                ASK NewTransporter TO SetDestination(Unit);
                ASK NewTransporter TO SetPosition(Unit.Position);
                ASK NewTransporter TO SetPort(Unit);
                ASK BigTransporterQ TO Add(NewTransporter);

                CASE ASK NewTransporter Class
                        WHEN Aircraft :
                                NewPort := Unit.AirPort;

                        WHEN Ship :
                                NewPort := Unit.SeaPort;

                        WHEN Rail :
                                NewPort := Unit.RailYard;

                        WHEN Truck :
                                NewPort := Unit.TruckStop;
                END CASE;
                ASK NewPort.ParkedQ TO Add(NewTransporter);
                ASK TransporterManager TO
                                    ReceiveAvailableTransporter(NewTransporter);
        END FOR;
END FOR;
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numitems := integer;
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR m := 1 TO numitems

        ASK File TO ReadString(string);
```

```
          Commodity := ASK CommodityQ TO FindByName(string);
          ASK File TO ReadLine(string);
          IF Commodity <> NILOBJ
                  NewCommodity := CLONE(Commodity);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetHighRate(real);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetMedRate(real);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetLowRate(real);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetNoneRate(real);
                  ASK File TO ReadLine(string);

                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetOnHand(real);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetStockTo(real);
                  ASK File TO ReadLine(string);

                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetOrderAt(real);
                  ASK File TO ReadString(string);
                  ASK File TO ReadReal(real);
                  ASK NewCommodity TO SetEmerOrderAt(real);
                  ASK File TO ReadLine(string);

                  ASK File TO ReadString(string);
                  ASK File TO ReadString(string);
                  ASK NewCommodity TO SetDeployment(string);
                  ASK File TO ReadLine(string);


                  ASK Unit.Inventory TO Add(NewCommodity)
          ELSE
                  OUTPUT("Unlisted Commodity in Unit file");
                  ASK File TO ReadLine(string);
                  ASK File TO ReadLine(string);
                  ASK File TO ReadLine(string);
                  ASK File TO ReadLine(string);
          END IF;
          ASK File TO ReadLine(string);
  END FOR;

  ASK File TO Close;
  DISPOSE(File);
  RETURN(Unit);

  END METHOD;

  {-------------------------------------------------------------------------}
  ASK METHOD BuildSupply;
```

```
{------------------------------------------------------------------------}

VAR
BEGIN

{NEW a supplyobj, Set its inventory to the commodity list, start production.}

NEW(Supply);
ASK Supply TO SetInventory(CommodityQ);
TELL Supply TO Produce;
ASK BaseQ TO Add(Supply);
ASK LogisticsManager.ConusQ TO Add(Supply);
END METHOD;

{------------------------------------------------------------------------}
ASK METHOD BuildScenario(IN FileName : STRING);
{------------------------------------------------------------------------}

{builds a scenario from scenario datafiles}

VAR

File : StreamObj;
string : STRING;
integer : INTEGER;
n, numItems : INTEGER;

BEGIN

NEW(File);

NEW(TransporterManager);

ASK File TO Open(FileName, Input);
ASK File TO ReadString(string);
ASK SELF TO BuildScenarioCommodities(string, CommodityQ);
ASK SELF TO BuildSupply;
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK SELF TO BuildScenarioTransporters(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK SELF TO BuildScenarioBases(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK SELF TO BuildScenarioUnits(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK SELF TO LinkUnits(string);
ASK File TO ReadLine(string);


ASK File TO ReadString(string);
ASK SELF TO LinkRailRoads(string);
ASK File TO ReadLine(string);
ASK File TO ReadString(string);
ASK SELF TO LinkRoads(string);

ASK File TO ReadString(string);
ASK SELF TO BuildScenarioPrepos(string);

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD BuildPrepo(IN FileName : STRING);
{-----------------------------------------------------------------------}

{builds a Prepo from Prepo datafile}

CONST

VAR

File : StreamObj;
string : STRING;
i, integer : INTEGER;
position : PositionRecType;
Transporter, NewTransporter : TransporterObj;
shipment, NewShipment : ShipmentObj;
commodity, NewCommodity : CommodityObj;
base, NextStop, Destination : BaseObj;
Route : BaseQObj;

Name : STRING;

BEGIN
{
OUTPUT("IN BUILD PREPO");
}
NEW(File);
NEW(shipment);

{Open the data file to be read}
ASK File TO Open(FileName, Input);
ASK File TO ReadLine(string);

{read prepo transporter name}
ASK File TO ReadString(Name);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

{read transporter name and find in the transporter Queue}
ASK File TO ReadString(string);
ASK File TO ReadString(string);
Transporter := ASK TransporterQ TO FindByName(string);
IF Transporter = NILOBJ
        OUTPUT("INVALID PREPO FILE - CANNOT FIND TRANSPORTER - ", Name);
        RETURN;
END IF;
NewTransporter := CLONE(Transporter);
ASK NewTransporter TO SetName(Name);
ASK Transporter TO SetVehicleID(Transporter.VehicleID + 1);
ASK NewTransporter TO SetVehicleID(Transporter.VehicleID);
ASK File TO ReadLine(string);
```

```
{read transporter position data}
NEW(position);
ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
position.LatDeg := integer;
ASK File TO ReadInt(integer);
position.LatMin := integer;
ASK File TO ReadString(string);
position.LatDir := string;
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
position.LongDeg := integer;
ASK File TO ReadInt(integer);
position.LongMin := integer;
ASK File TO ReadString(string);
position.LongDir := string;
ASK NewTransporter TO SetPosition(position);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

{Read and build shipment destination and cargo routing.}
ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK File TO ReadString(string);
base := ASK BaseQ TO FindByName(string);
IF base = NILOBJ
        OUTPUT("INVALID PREPO FILE - CANNOT FIND DESTINATION - ", Name);
        RETURN;
END IF;
ASK shipment TO SetDestination(base);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

FOR i := 1 TO integer
        ASK File TO ReadString(string);
        base := ASK BaseQ TO FindByName(string);
        IF base = NILOBJ
                OUTPUT("INVALID PREPO FILE - CANNOT FIND BASE - ", Name);
                RETURN;
        END IF;
        ASK shipment.Route TO Add(base);
        ASK File TO ReadLine(string);
END FOR;
NextStop := ASK shipment.Route First;
ASK NewTransporter TO SetDestination(NextStop);
ASK File TO ReadLine(string);

{Read and build cargo items.  Build each commodity, set onhand, place in shipmen
ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);
```

```
FOR i := 1 TO integer
        ASK File TO ReadString(string);
        commodity := ASK CommodityQ TO FindByName(string);
        IF commodity <> NILOBJ
                ASK File TO ReadReal(real);
                NewCommodity := CLONE(commodity);
                ASK NewCommodity TO SetOnHand(real);
                NEW(NewShipment);
                ASK NewShipment TO SetItem(NewCommodity);
                Route := ASK shipment Route;
                ASK NewShipment TO SetRoute(Route);
                Destination := ASK shipment Destination;
                ASK NewShipment TO SetDestination(Destination);
                ASK NewTransporter.Cargo TO Add(NewShipment);
        END IF;
        ASK File TO ReadLine(string);
END FOR;
ASK BigTransporterQ TO Add(NewTransporter);
ASK PrepoTransporterQ TO Add(NewTransporter);

ASK File TO Close;
DISPOSE(File);
DISPOSE(shipment);

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD BuildScenarioPrepos(IN FileName : STRING);
{---------------------------------------------------------------------------}

{builds all prepos in scenario}

VAR

File : StreamObj;
transporter : TransporterObj;
string : STRING;
integer : INTEGER;
n, numItems : INTEGER;


BEGIN
{
OUTPUT("IN BUILD SCENARIOPREPOS");
}
NEW(File);
ASK File TO Open(FileName, Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;

ASK File TO ReadLine(string);

FOR n := 1 TO numItems
        ASK File TO ReadString(string);
        string := SUBSTR(1,8,string) + ".dat";
        ASK SELF TO BuildPrepo(string);
END FOR;
ASK File TO Close;
```

```
DISPOSE(File);

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD BuildScenarioTransporters(IN FileName : STRING);
{---------------------------------------------------------------------}

{builds all Transporter in scenario}

VAR

File : StreamObj;
transporter : TransporterObj;
string : STRING;
integer : INTEGER;
n, numItems : INTEGER;


BEGIN

NEW(File);
ASK File TO Open(FileName, Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;

ASK File TO ReadLine(string);

FOR n := 1 TO numItems
        ASK File TO ReadString(string);
        transporter := ASK SELF TO BuildTransporter(string);
        ASK TransporterQ TO Add(transporter);
END FOR;
ASK File TO Close;
DISPOSE(File);

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD BuildScenarioBases(IN FileName : STRING);
{---------------------------------------------------------------------}

{builds all bases in scenario}

VAR

File : StreamObj;
base : BaseObj;
string : STRING;
integer : INTEGER;
n, numItems : INTEGER;


BEGIN

NEW(File);
ASK File TO Open(FileName, Input);
ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;

ASK File TO ReadLine(string);

FOR n := 1 TO numItems
        ASK File TO ReadString(string);
        string := SUBSTR(1,8,string);
        string := string + ".dat";
        base := ASK SELF TO BuildBase(string);
        TELL base TO CheckInventory;
        ASK BaseQ TO Add(base);
        ASK OnlyBaseQ TO Add(base);
        CASE base.Group
        WHEN CONUS:
                ASK LogisticsManager.ConusQ TO Add(base);
        WHEN ILOC:
                ASK LogisticsManager.ILocQ TO Add(base);
        WHEN THEATER:
                ASK LogisticsManager.TheaterQ TO Add(base);
        OTHERWISE
                ASK LogisticsManager.TheaterQ TO Add(base);
        END CASE;
        CASE base.SubGroup
        WHEN POE:
                ASK LogisticsManager.POEQ TO Add(base);
        WHEN POS:
                ASK LogisticsManager.POSQ TO Add(base);
        WHEN POD:
                ASK LogisticsManager.PODQ TO Add(base);
                ASK LogisticsManager.POSQ TO Add(base);
        OTHERWISE
        END CASE;

END FOR;
ASK File TO Close;
DISPOSE(File);

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD BuildScenarioUnits(IN FileName : STRING);
{---------------------------------------------------------------------------}

{builds all units in scenario}

VAR

File : StreamObj;
unit : UnitObj;
string : STRING;
real : REAL;
integer : INTEGER;
n, numItems : INTEGER;


BEGIN

NEW(File);
```

```
ASK File TO Open(FileName, Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;

ASK File TO ReadLine(string);

FOR n := 1 TO numItems
        ASK File TO ReadString(string);
        string := SUBSTR(1,8,string);
        string := string + ".dat";
        unit := ASK SELF TO BuildUnit(string);
        TELL unit TO {Consume} DelayActivation;
        ASK BaseQ TO Add(unit);
        ASK UnitQ TO Add(unit);
        ASK LogisticsManager.UnitQ TO Add(unit);

END FOR;
ASK File TO Close;
DISPOSE(File);

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD LinkUnits(IN FileName : STRING);
{--------------------------------------------------------------------------}

{links units to their Origins}

VAR

File : StreamObj;
string : STRING;
real : REAL;
integer : INTEGER;
n, numItems : INTEGER;
CurrentUnit : UnitObj;
LinkToBase : BaseObj;

BEGIN

{
OUTPUT("in Link Units");
}
NEW(File);
ASK File TO Open(FileName, Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;

ASK File TO ReadLine(string);
ASK File TO ReadLine(string);

FOR n := 1 TO numItems;

        ASK File TO ReadString(string);
        ASK File TO ReadString(string);
        CurrentUnit := ASK UnitQ TO FindByName(string);
```

```
         ASK File TO ReadLine(string);

         ASK File TO ReadString(string);
         ASK File TO ReadString(string);
         LinkToBase := ASK BaseQ TO FindByName(string);
         IF LinkToBase <> NILOBJ
                 ASK CurrentUnit TO SetOrigin(LinkToBase);
                 ASK CurrentUnit TO SetLinked;
         END IF;
         ASK File TO ReadLine(string);
         ASK File TO ReadLine(string);
END FOR;

END METHOD;

{-------------------------------------------------------------------------------}
     ASK METHOD LinkRailRoads(IN FileName : STRING);
{-------------------------------------------------------------------------------}

{links rail network}

VAR

File : StreamObj;
string : STRING;
real : REAL;
integer : INTEGER;
n, numItems : INTEGER;
m, numNodes : INTEGER;
CurrentBase : BaseObj;
LinkToBase : BaseObj;

BEGIN
{
OUTPUT("IN LinkRailRoads ");
}

NEW(File);
ASK File TO Open(FileName, Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR n := 1 TO numItems;

        ASK File TO ReadString(string);
        ASK File TO ReadString(string);
        CurrentBase := ASK BaseQ TO FindByName(string);
        ASK File TO ReadLine(string);

        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        numNodes := integer;
        ASK File TO ReadLine(string);

        ASK File TO ReadLine(string);
        FOR m := 1 TO numNodes;
                ASK File TO ReadString(string);
                LinkToBase := ASK BaseQ TO FindByName(string);
                IF LinkToBase <> NILOBJ
                        ASK CurrentBase.RailYard.Network TO Add(LinkToBase);
                END IF;
                ASK File TO ReadLine(string);
        END FOR;

        ASK File TO ReadLine(string);

END FOR;

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD LinkRoads(IN FileName : STRING);
{---------------------------------------------------------------------------}

{links scenario road network}

VAR

File : StreamObj;
string : STRING;
real : REAL;
integer : INTEGER;
n, numItems : INTEGER;
m, numNodes : INTEGER;
CurrentBase : BaseObj;
LinkToBase : BaseObj;

BEGIN
{
OUTPUT("IN LinkRoads ");
}

NEW(File);
ASK File TO Open(FileName, Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
numItems := integer;

ASK File TO ReadLine(string);
ASK File TO ReadLine(string);

FOR n := 1 TO numItems;

        ASK File TO ReadString(string);
        ASK File TO ReadString(string);
        CurrentBase := ASK BaseQ TO FindByName(string);
        ASK File TO ReadLine(string);

        ASK File TO ReadString(string);
        ASK File TO ReadInt(integer);
        numNodes := integer;

        ASK File TO ReadLine(string);
```

```
                ASK File TO ReadLine(string);


                FOR m := 1 TO numNodes;
                        ASK File TO ReadString(string);
                        LinkToBase := ASK BaseQ TO FindByName(string);
                        IF LinkToBase <> NILOBJ
                                ASK CurrentBase.TruckStop.Network TO Add(LinkToBase);
                        END IF;
                        ASK File TO ReadLine(string);

                END FOR;

        ASK File TO ReadLine(string);

        END FOR;


        END METHOD;

        {---------------------------------------------------------------------------}
             ASK METHOD ObjInit;
        {---------------------------------------------------------------------------}
        VAR
        BEGIN

        NEW(LogisticsManager);
        NEW(BaseQ);
        NEW(OnlyBaseQ);
        NEW(UnitQ);
        NEW(TransporterQ);
        NEW(BigTransporterQ);
        NEW(PrepoTransporterQ);
        NEW(CommodityQ);

        END METHOD;

        {---------------------------------------------------------------------------}
             ASK METHOD ObjTerminate;
        {---------------------------------------------------------------------------}
        VAR

        i, k, numItems, numItems2 : INTEGER;
        base : BaseObj;
        transporter : TransporterObj;
        shipment : ShipmentObj;

        BEGIN

        DISPOSE(LogisticsManager);
        numItems := ASK BaseQ numberIn;
        FOR i := 1 TO numItems
                base := ASK BaseQ TO Remove;
                ASK BaseQ TO Add(base);
                ASK base TO DisposePorts;
                {
                IF OBJTYPENAME(base) <> "SupplyObj"

                        numItems2 := ASK base.RailYard.Network numberIn;
                FOR k := 1 TO numItems2
                        ASK base.RailYard.Network TO RemoveThis(ASK
                                                base.RailYard.Network First);
                END FOR;
                numItems2 := ASK base.TruckStop.Network numberIn;
                FOR k := 1 TO numItems2
                        ASK base.TruckStop.Network TO RemoveThis(ASK
                                                base.TruckStop.Network First);
                END FOR;
                ASK base TC DisposePorts;
        END IF;
        }
END FOR;

IF ASK BaseQ Includes(Supply)

        ASK BaseQ TO RemoveThis(Supply);
        IF NumActivities(Supply) > 0
                InterruptAll(Supply);
                ASK TrashCan TO Add(Supply);
        ELSE
                DISPOSE(Supply);
        END IF;
END IF;

DISPOSE(BaseQ);
DISPOSE(OnlyBaseQ);
DISPOSE(UnitQ);
DISPOSE(TransporterQ);
numItems := ASK BigTransporterQ numberIn;
FOR i := 1 TO numItems
        transporter := ASK BigTransporterQ TO Remove;
        ASK BigTransporterQ TO Add(transporter);
        numItems2 := ASK transporter.Cargo numberIn;
        FOR k := 1 TO numItems2
                shipment := ASK transporter.Cargo TO Remove;
                DISPOSE(shipment);
        END FOR;
END FOR;

DISPOSE(BigTransporterQ);
DISPOSE(PrepoTransporterQ);
DISPOSE(CommodityQ);
END METHOD;

END OBJECT;

END MODULE.
```

```
DEFINITION MODULE CommodQ;

{CommodityObj, derived from the NamedObj, is the object that is moved between
bases.  The CommodityQObj is a refinment of the MyQueueObj which can only take
CommodityObjs as members.}

{Import statements}

FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;

{Type Declarations}

TYPE

CommodityClassType = (Fuel, FFV, Ammo, Spares, Personnel, Medical, Major,
                                                                    Other);

{============================================================================}
CommodityObj = OBJECT(NamedObj)
{============================================================================}

{Fields}

     Class : CommodityClassType;
     HighRate : REAL;
     MedRate : REAL;
     LowRate : REAL;
     NoneRate : REAL;

     OnHand : REAL;
     OnOrder : REAL;
     StockTo : REAL;
     OrderAt : REAL;
     Priority : INTEGER;
     NormalPriority : INTEGER;
     EmerOrderAt : REAL;
     EmerPriority : INTEGER;

     ProduceAt : REAL;
     Length : REAL;
     Width : REAL;
     Height : REAL;
     Weight : REAL;
     OverSize : BOOLEAN;
     Deployment : BOOLEAN;
     InPlace : BOOLEAN;

{Methods}
     ASK METHOD SetClass(IN NewClass : STRING);
     ASK METHOD SetHighRate(IN NewHighRate : REAL);
     ASK METHOD SetMedRate(IN NewMedRate : REAL);
     ASK METHOD SetLowRate(IN NewLowRate : REAL);
     ASK METHOD SetNoneRate(IN NewNoneRate : REAL);

     ASK METHOD SetOnHand(IN NewOnHand : REAL);
     ASK METHOD SetOnOrder(IN NewOnOrder : REAL);
     ASK METHOD SetStockTo(IN NewStockTo : REAL);
     ASK METHOD SetOrderAt(IN NewOrderAt : REAL);
     ASK METHOD SetPriority(IN NewPriority : INTEGER);
```

```
      ASK METHOD SetNormalPriority(IN NewNormalPriority : INTEGER);
      ASK METHOD SetEmerOrderAt(IN NewEmerOrderAt : REAL);
      ASK METHOD SetEmerPriority(IN NewEmerPriority : INTEGER);

      ASK METHOD SetProduceAt(IN NewProduceAt : REAL);

      ASK METHOD SetLength(IN NewLength : REAL);
      ASK METHOD SetWidth(IN NewWidth : REAL);
      ASK METHOD SetHeight(IN NewHeight : REAL);
      ASK METHOD SetWeight(IN NewWeight : REAL);
      ASK METHOD SetOverSize(IN NewOverSize : STRING);
      ASK METHOD SetDeployment(IN NewDeployment : STRING);
      ASK METHOD SetInPlace(IN NewInPlace : STRING);
      ASK METHOD AddOnHand(IN AddThis : REAL);
      ASK METHOD SubtractOnHand(IN MinusThis : REAL);
      ASK METHOD AddOnOrder(IN NewOnOrder : REAL);
      ASK METHOD SubtractOnOrder(IN AddThis : REAL);

      ASK METHOD Display(INOUT j : INTEGER);
      ASK METHOD Modify;
      ASK METHOD ObjInit;

END OBJECT;

{=======================================================================}
CommodityQObj = PROTO(MyQueueObj[NamedObj : #CommodityObj])
{=======================================================================}

OVERRIDE
ASK METHOD Display(INOUT j : INTEGER);
ASK METHOD ObjTerminate;

END PROTO;
END MODULE.
```

```
IMPLEMENTATION MODULE CommodQ;

{Commodity Object and Queue}

{Import statements}

FROM MyQueue IMPORT MyQueueObj,
                     NamedObj;
FROM Base IMPORT BaseObj;
FROM SOUTPUT IMPORT SOUTPUT;
FROM CRTMod IMPORT ClearScreen;
FROM IOMod IMPORT ReadKey;
{FROM IMPORT ;}

{Definitions}

{==============================================================================}
OBJECT CommodityObj;
{==============================================================================}

        {METHODS}

{------------------------------------------------------------------------------}
ASK METHOD SetClass(IN NewClass : STRING);
{------------------------------------------------------------------------------}
BEGIN

CASE NewClass
        WHEN "Fuel":
                Class := Fuel;
        WHEN "FFV" :
                Class := FFV;
        WHEN "Ammo":
                Class := Ammo;
        WHEN "Spares" :
                Class := Spares;
        WHEN "Personnel":
                Class := Personnel;
        WHEN "Medical":
                Class := Medical;
        WHEN "Major":
                Class := Major;
        OTHERWISE
                Class := Other;
END CASE;

END METHOD;

{------------------------------------------------------------------------------}
ASK METHOD SetHighRate(IN NewHighRate : REAL);
{------------------------------------------------------------------------------}
BEGIN

HighRate := NewHighRate;

END METHOD;

{------------------------------------------------------------------------------}
ASK METHOD SetMedRate(IN NewMedRate : REAL);
{------------------------------------------------------------------------------}
```

```
BEGIN

MedRate := NewMedRate;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetLowRate(IN NewLowRate : REAL);
{---------------------------------------------------------------------------}
BEGIN

LowRate := NewLowRate;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetNoneRate(IN NewNoneRate : REAL);
{---------------------------------------------------------------------------}
BEGIN

NoneRate := NewNoneRate;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetOnHand(IN NewOnHand : REAL);
{---------------------------------------------------------------------------}
BEGIN
          OnHand := NewOnHand;
          IF OnHand < 0.0
                OnHand := 0.0;
          END IF;
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetOnOrder(IN NewOnOrder : REAL);
{---------------------------------------------------------------------------}
BEGIN
          OnOrder := NewOnOrder;
          IF OnOrder < 0.0
                OnOrder := 0.0;
          END IF;
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetStockTo(IN NewStockTo : REAL);
{---------------------------------------------------------------------------}
BEGIN
          StockTo := NewStockTo;
END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD SetOrderAt(IN NewOrderAt : REAL);
{---------------------------------------------------------------------------}
BEGIN
          OrderAt := NewOrderAt;
END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD SetPriority(IN NewPriority : INTEGER);
```

```
{--------------------------------------------------------------------------}
BEGIN
          Priority := NewPriority;
END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetNormalPriority(IN NewNormalPriority : INTEGER);
{--------------------------------------------------------------------------}
BEGIN
          NormalPriority := NewNormalPriority;
END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetEmerOrderAt(IN NewEmerOrderAt : REAL); {--------------------
          EmerOrderAt := NewEmerOrderAt;
END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetEmerPriority(IN NewEmerPriority : INTEGER); {---------------
          EmerPriority := NewEmerPriority;
END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetProduceAt(IN NewProduceAt : REAL); {-----------------------
          ProduceAt := NewProduceAt;
END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetLength(IN NewLength : REAL);
{--------------------------------------------------------------------------}   B

        Length := NewLength;

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetWidth(IN NewWidth : REAL);
{--------------------------------------------------------------------------}   B

        Width := NewWidth;

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetHeight(IN NewHeight : REAL);
{--------------------------------------------------------------------------}   B

        Height := NewHeight;

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD SetWeight(IN NewWeight : REAL);
{--------------------------------------------------------------------------}   B

        Weight := NewWeight;

END METHOD;

{--------------------------------------------------------------------------}
```

```
      ASK METHOD SetOverSize(IN NewOverSize : STRING);
{------------------------------------------------------------------------}   B

IF NewOverSize = "TRUE"
        OverSize := TRUE;
ELSE
        OverSize := FALSE;
END IF;

END METHOD;

{------------------------------------------------------------------------}
      ASK METHOD SetDeployment(IN NewDeployment : STRING);
{------------------------------------------------------------------------}   B

IF NewDeployment = "TRUE"
        Deployment := TRUE;
ELSE
        Deployment := FALSE;
END IF;

END METHOD;

{------------------------------------------------------------------------}
      ASK METHOD SetInPlace(IN NewInPlace : STRING);
{------------------------------------------------------------------------}   B

IF NewInPlace = "TRUE"
        InPlace := TRUE;
ELSE
        InPlace := FALSE;
END IF;

END METHOD;

{------------------------------------------------------------------------}
      ASK METHOD AddOnHand(IN AddThis : REAL);
{------------------------------------------------------------------------}
BEGIN
        OnHand := OnHand + AddThis;
          IF OnHand < 0.0
                OnHand := 0.0;
          END IF;
END METHOD;

{------------------------------------------------------------------------}
      ASK METHOD SubtractOnHand(IN MinusThis : REAL);
{------------------------------------------------------------------------}
BEGIN
          OnHand := OnHand - MinusThis;
          IF OnHand < 0.0
                OnHand := 0.0;
          END IF;
END METHOD;

{------------------------------------------------------------------------}
ASK METHOD AddOnOrder(IN AddThis : REAL);
{------------------------------------------------------------------------}
BEGIN
          OnOrder := OnOrder + AddThis;
```

```
            IF OnOrder < 0.0
                  OnOrder := 0.0;
            END IF;
END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD SubtractOnOrder(IN MinusThis : REAL);
{--------------------------------------------------------------------------}
BEGIN
        OnOrder := OnOrder - MinusThis;
          IF OnOrder < 0.0
                OnOrder := 0.0;
          END IF;
END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD Display(INOUT j : INTEGER);
{--------------------------------------------------------------------------}
CONST

format = "**************** ********* **** x *** x *** ********* ********** ******"
 + "        **";

VAR

string : STRING;

BEGIN

string := SPRINT(Name, Class, Length, Width, Height, Weight, ProduceAt,
                                                    OverSize, Priority) WITH format;
SOUTPUT(string,j);

END METHOD;


{--------------------------------------------------------------------------}
ASK METHOD Modify;
{--------------------------------------------------------------------------}
VAR

CHR : CHAR;
string : STRING;
real : REAL;
j, integer : INTEGER;


BEGIN

LOOP
        ClearScreen;
        OUTPUT(" ");
        SOUTPUT("Name              Class        Dim. (inches)    Weight" +
 "  Prod Rate O/S   Priority",j);
OUTPUT("=====================================================================

ASK SELF TO Display(j);
OUTPUT("");
OUTPUT("     Modify? (N)ame, (C)lass, (D)imensions, (W)ght, ProdRa(t)e," +
" (P)ri, (R)eturn");
CHR := ReadKey();

IF (CHR = "N") OR (CHR = "n")
        ASK SELF TO InputName;
ELSIF (CHR = "C") OR (CHR = "c")
        LOOP
                OUTPUT("     Input New Commodity Class:  ");
                OUTPUT("     (F)uel, FF(V), (A)mmo, (S)pares, (P)ersonnel, (M)ed
                CHR := ReadKey();
                IF (CHR = "F") OR (CHR = "f")
                        ASK SELF TO SetClass("Fuel");
                        EXIT;
                ELSIF  (CHR = "V") OR (CHR = "v")
                        ASK SELF TO SetClass("FFV");
                        EXIT;
                ELSIF  (CHR = "A") OR (CHR = "a")
                        ASK SELF TO SetClass("Ammo");
                        EXIT;
                ELSIF  (CHR = "S") OR (CHR = "s")
                        ASK SELF TO SetClass("Spares");
                        EXIT;
                ELSIF  (CHR = "P") OR (CHR = "p")
                        ASK SELF TO SetClass("Personnel");
                        EXIT;
                ELSIF  (CHR = "M") OR (CHR = "m")
                        ASK SELF TO SetClass("Medical");
                        EXIT;
                ELSIF  (CHR = "R") OR (CHR = "r")
                        ASK SELF TO SetClass("Major");
                        EXIT;
                ELSIF  (CHR = "O") OR (CHR = "o")
                        ASK SELF TO SetClass("Other");
                        EXIT;
                END IF;
        END LOOP;
ELSIF (CHR = "T") OR (CHR = "t")
        OUTPUT("     Input Commodity Production Rate (REAL numbers)");
        INPUT(real);
        ASK SELF TO SetProduceAt(real);
ELSIF (CHR = "D") OR (CHR = "d")
OUTPUT("     Input Commodity Length (REAL inches)");
INPUT(real);
ASK SELF TO SetLength(real);

OUTPUT("     Input Commodity Width (REAL inches)");
INPUT(real);
ASK SELF TO SetWidth(real);

OUTPUT("     Input Commodity Height (REAL inches)");
INPUT(real);
ASK SELF TO SetHeight(real);

ELSIF (CHR = "W") OR (CHR = "w")

OUTPUT("     Input Commodity Weight (REAL inches)");
INPUT(real);
ASK SELF TO SetWeight(real);
```

```
ELSIF (CHR = "P") OR (CHR = "p")
LOOP
        OUTPUT("     (N)ormal or (E)mergency");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n")
                LOOP
                        OUTPUT("     Current Priority is ", NormalPriority);
                        OUTPUT("     Input New Priority (1-12)");
                        INPUT(integer);
                        IF (integer > 0) AND (integer < 13)
                                ASK SELF TO SetNormalPriority(integer);
                                EXIT;
                        END IF;
                END LOOP;
                EXIT;
        ELSIF (CHR = "E") OR (CHR = "e")
                LOOP
                        OUTPUT("     Current Emergency Priority is ",
                                 EmerPriority);
                        OUTPUT("     Input New Emergency Priority (1-12)");
                        INPUT(integer);
                        IF (integer > 0) AND (integer < 13)
                                ASK SELF TO SetEmerPriority(integer);
                                EXIT;
                        END IF;
                END LOOP;
                EXIT;
        END IF;
END LOOP;
IF (SELF.Length > 1090.0) OR (SELF.Width > 117.0) OR
                                                    (SELF.Height > 105.0)
        ASK SELF TO SetOverSize("TRUE");
ELSE
        ASK SELF TO SetOverSize("FALSE");
END IF;
ELSIF (CHR = "R") OR (CHR = "r")
        EXIT;
END IF;

END LOOP;
END METHOD;



{-------------------------------------------------------------------------}
ASK METHOD ObjInit; {----------------------------------------------------------
BEGIN


END METHOD;


END OBJECT;

{=========================================================================}
PROTO CommodityQObj;
{=========================================================================}
```

```
{--------------------------------------------------------------------------}
ASK METHOD Display(INOUT j :INTEGER);
{--------------------------------------------------------------------------}

VAR

string : STRING;
Commodity : CommodityObj;
ID, i, numItems : INTEGER;

BEGIN

        ClearScreen;
        SOUTPUT(" ",j);
        SOUTPUT("Name              Class        Dim. (inches)    Weight" +
 "  Prod Rate O/S   Priority",j);
OUTPUT("================================================================================
        SOUTPUT(" ",j);
        numItems := numberIn;
        FOR i := 1 TO numItems
                Commodity := ASK SELF TO Remove;
                ASK SELF TO Add(Commodity);
                ASK Commodity TO Display(j);
        END FOR;
        SOUTPUT(" ",j);
SOUTPUT("================================================================================

END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD ObjTerminate; {-------------------------------------------------
VAR

i, numItems : INTEGER;
object : CommodityObj;

BEGIN

numItems := numberIn;
FOR i := 1 TO numItems
        object := ASK SELF TO Remove;
        DISPOSE(object);
END FOR;

END METHOD;




END PROTO;

END MODULE.
```

```
DEFINITION MODULE DOSMenu;

{Comments}

{Import statements}

FROM Trash IMPORT TrashCan,
                  GarbageDisposal;

FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;

FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj,
                    CommodityClassType;

FROM Trnsprt IMPORT TransporterObj,
                    TransporterQObj,
                    TransporterClassType;

FROM Base IMPORT BaseObj,
                 BaseQObj;

FROM Port IMPORT PortObj,
                 CargoGroupObj,
                 LoadingDockObj;

FROM Distant IMPORT PositionRecType,
                    CalcDistance;

FROM Builder IMPORT Builder;
FROM LogMan IMPORT LogisticsManager;

{Type Declarations}

TYPE

{======================================================================}
DOSMenuObj  = OBJECT; {===================================================

{DOSMenu Object displays menus and controls program execution}

{FIELDS}
Scenario : NamedObj;
ScenarioList : MyQueueObj;
{METHODS}

ASK METHOD DisplayMainMenu;

ASK METHOD DisplayScenarioMenu;
ASK METHOD DisplayBuildCommodityMenu;
ASK METHOD DisplayBuildTransporterMenu;
ASK METHOD DisplayBuildBaseMenu;
ASK METHOD DisplayBuildSubUnitMenu;
ASK METHOD DisplayBuildUnitMenu;
ASK METHOD DisplayBuildScenarioMenu;
ASK METHOD DisplayBuildPrepoMenu;

ASK METHOD DisplayScenarioSelectionMenu;
ASK METHOD DisplayExecutionMenu;
ASK METHOD SetUpScenario(IN ScenarioName : STRING);
TELL METHOD RunScenario(IN ScenarioName : STRING);
ASK METHOD DisplayTimeStepMenu(IN ScenarioName : STRING; INOUT TimeStep : REAL;
                                                          INOUT StopSim : BOOLEAN);
ASK METHOD DisplayBaseDisplayMenu;
ASK METHOD DisplayBase(IN Name : STRING);
ASK METHOD DisplayAllBases;
ASK METHOD ModifyBase(IN Name : STRING);
ASK METHOD DisplayUnitDisplayMenu;
ASK METHOD DisplayUnit(IN Name : STRING);
ASK METHOD DisplayAllUnits;
ASK METHOD ModifyUnit(IN Name : STRING);
ASK METHOD DisplayTransporterDisplayMenu;
ASK METHOD DisplayTransporter(IN Name : STRING; IN VehicleID : INTEGER);
ASK METHOD DisplayPrepoMenu;
ASK METHOD DisplayAllTransporters;
ASK METHOD ModifyTransporter;
ASK METHOD DisplayCommodityDisplayMenu;
ASK METHOD DisplayCommodity(IN Name : STRING);

ASK METHOD DisplayHelpMenu;

ASK METHOD DisplayOverView(IN BaseQ : BaseQObj);
ASK METHOD DisplayDeploymentOverView(IN BaseQ : BaseQObj);
ASK METHOD DisplayBaseOverView(IN Base : BaseObj);
ASK METHOD DisplayBaseDeploymentOverView(IN Base : BaseObj);

{x} ASK METHOD ReadScenarioMasterFile;

END OBJECT;

VAR

DOSMenu : DOSMenuObj;

END MODULE.
```

```
IMPLEMENTATION MODULE DOSMenu;

{Comments}

{Import statements}
FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj,
                    ALL CommodityClassType;

FROM Trnsprt IMPORT TransporterObj,
                    TransporterQObj,
                    ALL TransporterClassType;
FROM TManage IMPORT TransporterManager;
FROM Base IMPORT BaseObj,
                 BaseQObj;

FROM Port IMPORT PortObj,
                 CargoGroupObj,
                 LoadingDockObj;

FROM Distant IMPORT PositionRecType,
                    CalcDistance;
FROM CRTMod IMPORT ClearScreen;
FROM IOMod IMPORT StreamObj,
                  ALL FileUseType,
                  ReadKey;
FROM SimMod IMPORT StartSimulation,
                   StopSimulation,
                   SimTime,
                   ResetSimTime,
                   InterruptAll;
FROM Unit IMPORT UnitObj,
                 ALL UnitClassType,
                 ALL CombatIntensityType;
FROM SubUnit IMPORT SubUnitObj,
                    SubUnitQObj;
FROM Builder IMPORT Builder;
FROM ScenEd IMPORT ScenarioEditor;
FROM Scene IMPORT ScenarioObj;
FROM Trash IMPORT TrashCan,
                  GarbageDisposal;
FROM SOUTPUT IMPORT SOUTPUT;
FROM Stats IMPORT StatObj;
FROM Shpmnt IMPORT ShipmentObj;
FROM LogMan IMPORT LogisticsManager;
FROM DeBug IMPORT DebugBreak;
{
FROM ScenEdt IMPORT ScenarioEditor;
}

{Definitions}

{===========================================================================}
OBJECT DOSMenuObj;
{===========================================================================}

     {METHODS}
{---------------------------------------------------------------------------}
```

```
ASK METHOD DisplayMainMenu;
{---------------------------------------------------------------------------}
VAR

Answer : STRING;
CHR : CHAR;

BEGIN
NEW(TrashCan);
NEW(GarbageDisposal);
LOOP
        DebugBreak();
        ClearScreen;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                                    MAIN MENU");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                                1.  (S)cenario Menu. ");
        OUTPUT("                                2.  (E)xecution Menu. ");
        OUTPUT(" ");
        OUTPUT("                                3.  (H)elp Menu. ");
        OUTPUT("                                4.  (Q)uit.");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
        CHR := ReadKey();
        IF (CHR = "S") OR (CHR = "s") OR (CHR = "1")
                OUTPUT("     DISPLAYING SCENARIO MENU.");
                ASK SELF TO DisplayScenarioMenu;
        ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
                OUTPUT("     DISPLAYING EXECUTION MENU.");
                ASK SELF TO DisplayExecutionMenu;
        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "Q") OR (CHR = "q") OR (CHR = "4")
                OUTPUT("     QUITTING PROGRAM");
                ClearScreen;
                HALT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;
END LOOP;

END METHOD;

{-------------------------------------------------------------------------}
ASK METHOD DisplayScenarioMenu;
{-------------------------------------------------------------------------}
VAR

Answer : STRING;
CHR : CHAR;

BEGIN

NEW(TransporterManager);
NEW(ScenarioEditor);

LOOP
        DebugBreak();
        ClearScreen;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                                    SCENARIO MENU");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                              1.  (S)cenario Builder. ");
        OUTPUT("                              2.  (U)nit Builder. ");
        OUTPUT("                              3.  (B)ase Builder. ");
        OUTPUT("                              4.  (T)ransporter Builder.");
        OUTPUT("                              5.  (C)ommodity Builder.");
        OUTPUT("                              6.  SubU(n)it Builder.");
        OUTPUT("                              7.  (P)repo Builder");
        OUTPUT(" ");
        OUTPUT("                              8.  (H)elp Menu. ");
        OUTPUT("                              9.  (R)eturn to Main Menu. ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("      ENTER SELECTION AND STRIKE <ENTER>. ");
        CHR := ReadKey();
        IF (CHR = "S") OR (CHR = "s") OR (CHR = "1")
                OUTPUT("     DISPLAYING SCENARIO BUILDER.");
                ASK SELF TO DisplayBuildScenarioMenu;
        ELSIF (CHR = "U") OR (CHR = "u") OR (CHR = "2")
                OUTPUT("     DISPLAYING UNIT BUILDER.");
                ASK SELF TO DisplayBuildUnitMenu;

        ELSIF (CHR = "B") OR (CHR = "b") OR (CHR = "3")
```

```
                OUTPUT("     DISPLAYING BASE BUILDER.");
                ASK SELF TO DisplayBuildBaseMenu;

        ELSIF (CHR = "T") OR (CHR = "t") OR (CHR = "4")

                OUTPUT("     DISPLAYING TRANSPORTER MENU.");
                ASK SELF TO DisplayBuildTransporterMenu;
        ELSIF (CHR = "C") OR (CHR = "c") OR (CHR = "5")

                OUTPUT("     DISPLAYING COMMODITY MENU.");
                ASK SELF TO DisplayBuildCommodityMenu;
        ELSIF (CHR = "N") OR (CHR = "n") OR (CHR = "6")

                OUTPUT("     DISPLAYING SUBUNIT MENU.");
                ASK SELF TO DisplayBuildSubUnitMenu;
        ELSIF (CHR = "P") OR (CHR = "p") OR (CHR = "7")

                OUTPUT("     DISPLAYING SUBUNIT MENU.");
                ASK SELF TO DisplayBuildPrepoMenu;
        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "8")

                OUTPUT("DISPLAYING HELP MENU.");
        ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "9")
                OUTPUT("     RETURNING TO MAIN MENU");
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

END LOOP;

DISPOSE(ScenarioEditor);
DISPOSE(TransporterManager);
DebugBreak();
ASK GarbageDisposal TO TakeOutTheGarbage;
DebugBreak();


END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD DisplayBuildCommodityMenu;
{-----------------------------------------------------------------------------}
VAR


j : INTEGER;
Answer : STRING;
CHR : CHAR;
string : STRING;
commodity : CommodityObj;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT("                                        COMMODITY BUILDER");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT("                                   1.  Build (N)ew Commodity. ");
OUTPUT("                                   2.  Modify (E)xisting Commodity. ");
OUTPUT(" ");
OUTPUT("                                   3.  (H)elp Menu. ");
OUTPUT("                                   4.  (R)eturn to Scenario Menu.");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
CHR := ReadKey();
IF (CHR = "N") OR (CHR = "n") OR (CHR = "1")
        OUTPUT("     BUILDING NEW COMMODITY.");

        ASK ScenarioEditor TO CreateCommodity;

ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
        OUTPUT("     DISPLAYING MASTER COMMODITYLIST.");

        LOOP
                ClearScreen;
                j := 0;
                ASK ScenarioEditor.CommodityQ TO Display(j);
                SOUTPUT("     Which Commodity do you wish to" +
                          " modify?",j);
                INPUT(string);
                commodity := ASK ScenarioEditor.CommodityQ TO
                        FindByName(string);
                IF commodity <> NILOBJ
                        ASK commodity TO Modify;
                        EXIT;
                END IF;
        END LOOP;

ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
        OUTPUT("     DISPLAYING HELP MENU.");

ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "4")
        OUTPUT("     RETURNING TO MAIN MENU.");
        ClearScreen;
        EXIT;
ELSE
        OUTPUT("INVALID INPUT");
        END IF;

    END LOOP;

    ASK ScenarioEditor TO SaveMasterCommodityFile;

    END METHOD;

    {----------------------------------------------------------------------}
    ASK METHOD DisplayBuildTransporterMenu;
    {----------------------------------------------------------------------}
    VAR

    j : INTEGER;
    Answer : STRING;
    CHR : CHAR;
    string : STRING;
    transporter : TransporterObj;

    BEGIN

    LOOP
            ClearScreen;
            j := 0;
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT("                                  TRANSPORTER BUILDER");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT("                              1.  Build (N)ew Transporter. ");
            OUTPUT("                              2.  Modify (E)xisting Transporter.")
            OUTPUT(" ");
            OUTPUT("                              3.  (H)elp Menu. ");
            OUTPUT("                              4.  (R)eturn to Scenario Menu.");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT(" ");
            OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
            CHR := ReadKey();
            IF (CHR = "N") OR (CHR = "n") OR (CHR = "1")
                    OUTPUT("     BUILDING NEW TRANSPORTER.");

                    ASK ScenarioEditor TO CreateTransporter;
        ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
                OUTPUT("     DISPLAYING MASTER TRANSPORTER LIST.");

                ClearScreen;
                j := 0;

                LOOP
                ASK ScenarioEditor.TransporterQ TO Display(j);
                SOUTPUT("     Which Transporter do you wish to modify?",j);
                INPUT(string);
                transporter := ASK ScenarioEditor.TransporterQ TO
                                                        FindByName(string);
                IF transporter <> NILOBJ
                        ASK transporter TO Modify;
                        EXIT;
                END IF;
                END LOOP;


        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "4")
                OUTPUT("     RETURNING TO MAIN MENU.");
                ClearScreen;
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

ASK ScenarioEditor TO SaveMasterTransporterFile;

END LOOP;


END METHOD;
{--------------------------------------------------------------------------}
ASK METHOD DisplayBuildBaseMenu;
{--------------------------------------------------------------------------}
VAR

j : INTEGER;
Answer : STRING;
CHR : CHAR;
base : BaseObj;
string : STRING;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                                    BASE BUILDER");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                               1.  Build (N)ew Base. ");
        OUTPUT("                               2.  Modify (E)xisting Base.");
        OUTPUT(" ");
        OUTPUT("                               3.  (H)elp Menu. ");
        OUTPUT("                               4.  (R)eturn to Scenario Menu.");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n") OR (CHR = "1")
                OUTPUT("     BUILDING NEW BASE.");

                ASK ScenarioEditor TO CreateBase;

        ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
                OUTPUT("     DISPLAYING MASTER BASE LIST.");
                ClearScreen;
                j:= 0;

                LOOP
                ASK ScenarioEditor.OnlyBaseQ TO Display(j);
                SOUTPUT("     Which Base do you wish to modify?",j);
                INPUT(string);
                base := ASK ScenarioEditor.OnlyBaseQ TO
                                                    FindByName(string);
                IF base <> NILOBJ
                        ASK base TO Modify(ScenarioEditor);
                        EXIT;
                END IF;
                END LOOP;

        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "4")
                OUTPUT("     RETURNING TO MAIN MENU.");
                ClearScreen;
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

ASK ScenarioEditor TO SaveMasterBaseFile;

END LOOP;
END METHOD;

{-------------------------------------------------------------------------}
ASK METHOD DisplayBuildUnitMenu;
{-------------------------------------------------------------------------}
VAR

j : INTEGER;
Answer : STRING;
CHR : CHAR;
unit : BaseObj;
string : STRING;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                              UNIT BUILDER");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                      1.  Build (N)ew Unit. ");
        OUTPUT("                      2.  Modify (E)xisting Unit.");
        OUTPUT(" ");
        OUTPUT("                      3.  (H)elp Menu. ");
        OUTPUT("                      4.  (R)eturn to Scenario Menu.");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n") OR (CHR = "1")
                OUTPUT("     BUILDING NEW UNIT.");

                ASK ScenarioEditor TO CreateUnit;

        ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
                OUTPUT("     DISPLAYING MASTER BASE LIST.");
                ClearScreen;
                j:= 0;
                LOOP
                ASK ScenarioEditor.UnitQ TO Display(j);
                SOUTPUT("     Which Unit do you wish to modify?",j);
                INPUT(string);
                unit := ASK ScenarioEditor.UnitQ TO FindByName(string);
                IF unit <> NILOBJ
                        ASK unit TO Modify(ScenarioEditor);
                        EXIT;
                END IF;
                END LOOP;

        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "4")
                OUTPUT("     RETURNING TO MAIN MENU.");
                ClearScreen;
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

ASK ScenarioEditor TO SaveMasterUnitFile;

END LOOP;


END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD DisplayBuildSubUnitMenu;
{---------------------------------------------------------------------------}
VAR
j : INTEGER;
Answer : STRING;
CHR : CHAR;
subunit : SubUnitObj;
SubUnitQ : SubUnitQObj;
string : STRING;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                            SUBUNIT BUILDER");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                       1.  Build (N)ew SubUnit. ");
        OUTPUT("                       2.  Modify (E)xisting SubUnit.");
OUTPUT(" ");
OUTPUT("                              3.  (H)elp Menu. ");
OUTPUT("                              4.  (R)eturn to Scenario Menu.");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT(" ");
OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
CHR := ReadKey();
IF (CHR = "N") OR (CHR = "n") OR (CHR = "1")
        OUTPUT("     BUILDING NEW SUBUNIT.");
        NEW(subunit);
        ASK subunit TO Create;
        LOOP
                ASK subunit TO Modify;
                OUTPUT("     Save SubUnit (Y or N)");
                CHR := ReadKey();
                IF  (CHR = "Y") OR (CHR = "y")
                        CASE subunit.Class
                                WHEN Air: SubUnitQ :=
                                         ScenarioEditor.Planes;
                                WHEN Sea: SubUnitQ :=
                                         ScenarioEditor.Ships;
                                WHEN Land: SubUnitQ :=
                                         ScenarioEditor.Tanks;
                        OTHERWISE
                        END CASE;
                        ASK SubUnitQ TO Add(subunit);
                        EXIT;
                ELSIF  (CHR = "N") OR (CHR = "n")
                        DISPOSE(subunit);
                        EXIT;
                END IF;
        END LOOP;

ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
        OUTPUT("     DISPLAYING MASTER SUBUNIT LIST.");
        LOOP
                OUTPUT("     (A)ir, (S)ea, (L)and, (R)eturn?");
                CHR := ReadKey();
                IF (CHR = "A") OR (CHR = "a")
                        SubUnitQ := ScenarioEditor.Planes;
                        EXIT;
                ELSIF (CHR = "S") OR (CHR = "s")
                        SubUnitQ :=  ScenarioEditor.Ships;
                        EXIT;
                ELSIF (CHR = "L") OR (CHR = "l")
                        SubUnitQ :=  ScenarioEditor.Tanks;
                        EXIT;

                ELSIF (CHR = "R") OR (CHR = "r")
                        EXIT;
                        END IF;
                END LOOP;
                ClearScreen;
                j := 0;

                IF SubUnitQ <> NILOBJ
                        ASK SubUnitQ TO Display(j);
                        SOUTPUT("     Which SubUnit do you wish to modify?",j);
                        INPUT(string);
                        subunit := ASK SubUnitQ TO FindByName(string);
                        IF subunit <> NILOBJ
                                ASK subunit TO Modify;
                        END IF;
                END IF;


        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "4")
                OUTPUT("     RETURNING TO MAIN MENU.");
                ClearScreen;
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;


END LOOP;

ASK ScenarioEditor TO SaveMasterSubUnitFile;


END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD DisplayBuildScenarioMenu;
{----------------------------------------------------------------------------}
VAR

j : INTEGER;
Answer : STRING;
CHR : CHAR;
scene : ScenarioObj;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
```

```
          OUTPUT(" ");
          OUTPUT("                                          SCENARIO BUILDER");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT("                                    1.  Build (N)ew Scenario. ");
          OUTPUT("                                    2.  Modify (E)xisting Scenario.");
          OUTPUT(" ");
          OUTPUT("                                    3.  (H)elp Menu. ");
          OUTPUT("                                    4.  (R)eturn to Scenario Menu.");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
          CHR := ReadKey();
          IF (CHR = "N") OR (CHR = "n") OR (CHR = "1")
                  OUTPUT("     BUILDING NEW SCENARIO.");

                  ASK ScenarioEditor TO CreateScenario;

          ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
                  OUTPUT("     DISPLAYING MASTER SCEANRIO LIST.");
                  ClearScreen;
                  j := 0;
                  ASK ScenarioEditor.ScenarioList TO Display(j);
                  OUTPUT("");
                  OUTPUT("     Input Scenario Name To Modify and hit <RETURN>.");
                  INPUT(Answer);
                  scene := ASK ScenarioEditor.ScenarioList TO FindByName(Answer);
                  IF scene <> NILOBJ
                          ASK scene TO Modify;
                  END IF;
          ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
                  OUTPUT("     DISPLAYING HELP MENU.");

          ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "4")
                  OUTPUT("     RETURNING TO MAIN MENU.");
                  ClearScreen;
                  EXIT;
          ELSE
                  OUTPUT("INVALID INPUT");
          END IF;

  END LOOP;

  ASK ScenarioEditor TO SaveScenarioMasterFile;

  END METHOD;

  {----------------------------------------------------------------------------}
  ASK METHOD DisplayBuildPrepoMenu;
  {----------------------------------------------------------------------------}
  VAR
```

```
j : INTEGER;
Answer : STRING;
CHR : CHAR;
prepo : NamedObj;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                                  PREPO BUILDER");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("                             1.  Build (N)ew Prepo. ");
        OUTPUT("                             2.  Modify (E)xisting Prepo.");
        OUTPUT(" ");
        OUTPUT("                             3.  (H)elp Menu. ");
        OUTPUT("                             4.  (R)eturn to Scenario Menu.");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n") OR (CHR = "1")
                OUTPUT("     BUILDING NEW PREPO.");

                ASK ScenarioEditor TO CreatePrepo;

        ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
                OUTPUT("     DISPLAYING MASTER PREPO LIST.");
                ClearScreen;
                j := 0;
                ASK ScenarioEditor.PrepoQ TO Display(j);
                OUTPUT("");
                OUTPUT("     Input Prepo To Modify and hit <RETURN>.");
                INPUT(Answer);
                prepo := ASK ScenarioEditor.PrepoQ TO FindByName(Answer);
                IF prepo <> NILOBJ
{                       ASK ScenarioEditor TO ModifyPrepo(prepo.Name);}
                END IF;
        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "3")
```

```
                    OUTPUT("      DISPLAYING HELP MENU.");

          ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "4")
                    OUTPUT("      RETURNING TO MAIN MENU.");
                    ClearScreen;
                    EXIT;
          ELSE
                    OUTPUT("INVALID INPUT");
          END IF;

END LOOP;

ASK ScenarioEditor TO SavePrepoMasterFile;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD DisplayExecutionMenu;
{-----------------------------------------------------------------------------}
VAR
object : NamedObj;
Answer : STRING;
CHR : CHAR;

ScenarioName : STRING;

BEGIN

NEW(ScenarioList);
ASK SELF TO ReadScenarioMasterFile;
Scenario := ASK ScenarioList First;

LOOP
          DebugBreak();
          ClearScreen;
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT("                                        EXECUTION MENU");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT("                                  1.  (S)elect Scenario. ");
          OUTPUT("                                  2.  (E)xecute Scenario. ");
          OUTPUT("                                  3.  Reset Scenario (C)lock.");
          OUTPUT(" ");
          OUTPUT("                                  4.  (H)elp Menu. ");
          OUTPUT("                                  5.  (R)eturn to Main Menu.");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
          OUTPUT(" ");
```

```
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT(" ");
        OUTPUT("     ENTER SELECTION AND STRIKE <ENTER>. ");
        CHR := ReadKey();
        IF (CHR = "S") OR (CHR = "s") OR (CHR = "1")
                OUTPUT("     DISPLAYING SCENARIO SELECTION MENU.");
                ASK SELF TO DisplayScenarioSelectionMenu;
        ELSIF (CHR = "E") OR (CHR = "e") OR (CHR = "2")
                OUTPUT("     DISPLAYING TIME STEP MENU.");
                ASK SELF TO SetUpScenario(Scenario.Name);
        ELSIF (CHR = "C") OR (CHR = "c") OR (CHR = "3")
                ResetSimTime(0.0);
        ELSIF (CHR = "H") OR (CHR = "h") OR (CHR = "4")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r") OR (CHR = "5")
                OUTPUT("     RETURNING TO MAIN MENU.");
                ClearScreen;
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

END LOOP;
DISPOSE(ScenarioList);
ASK GarbageDisposal TO TakeOutTheGarbage;
END METHOD;

{----------------------------------------------------------------------------}
     ASK METHOD DisplayScenarioSelectionMenu;
{----------------------------------------------------------------------------}

{allows user to choose scenario, called by DisplayExecutionMenu}

VAR

j : INTEGER;
string : STRING;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        ASK ScenarioList TO Display(j);
        OUTPUT("     Input Scenario Name.");
        INPUT(string);
        Scenario := ASK ScenarioList TO FindByName(string);
        IF Scenario <> NILOBJ
                EXIT;
        END IF;

END LOOP;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD SetUpScenario(IN ScenarioName : STRING);
```

```
{----------------------------------------------------------------------}

{sets up scenario, called from DisplayExecutionMenu}

VAR

FileName : STRING;

BEGIN

NEW(Builder);

ClearScreen;
FileName := ScenarioName + ".scn";
ASK Builder TO BuildScenario(FileName);
TELL SELF TO RunScenario(ScenarioName);
StartSimulation;

END METHOD;

{----------------------------------------------------------------------}
TELL METHOD RunScenario(IN ScenarioName : STRING);
{----------------------------------------------------------------------}

{THIS IS IT, THIS RUNS THE SCENARIO?!?}

VAR

Answer : STRING;
FileName : STRING;
TimeStep : REAL;
StopSim : BOOLEAN;

BEGIN
TimeStep := 24.0;
ASK SELF TO DisplayTimeStepMenu(ScenarioName,TimeStep,StopSim);

LOOP
        IF StopSim
                EXIT;
        END IF;
        IF TimeStep < 0.0
                TimeStep := 0.0
        END IF;
        WAIT DURATION TimeStep;
                ASK SELF TO DisplayTimeStepMenu(ScenarioName,TimeStep,StopSim);
        END WAIT;
END LOOP;

        StopSimulation;
        {ASK Builder TO Reset;}
        DISPOSE(Builder);
        DISPOSE(TransporterManager);
        ASK GarbageDisposal TO TakeOutTheGarbage;

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD DisplayTimeStepMenu(IN ScenarioName : STRING; INOUT TimeStep : REAL;
                                                  INOUT StopSim : BOOLEAN);
```

```
{----------------------------------------------------------------------------}
VAR

Answer : STRING;
CHR : CHAR;
Deployment : BOOLEAN;

BEGIN

LOOP
        DebugBreak();
        ClearScreen;
        OUTPUT(" ");
        IF Deployment
                OUTPUT("               DISPLAYING UNIT DEPLOYMENT STATUS AT TIME
                DisplayDeploymentOverView(Builder.UnitQ);
        ELSE
                OUTPUT("                 DISPLAYING UNIT SUPPLY STATUS AT TIME "
                DisplayOverView(Builder.UnitQ);
        END IF;
        OUTPUT("     EXECUTING SCENARIO - ", ScenarioName);
        OUTPUT("     Time Step is - ", TimeStep, " hours.");
        OUTPUT(" ");
        OUTPUT("     COMMAND:  (S)tart/Resume, (N)ew Time Step, (R)eturn/Stop");
        IF Deployment
                OUTPUT("     DISPLAY:  (B)ases, (U)nits, (T)ransporters," +
                         " (C)ommodities, (D)eploy OFF");
        ELSE
                OUTPUT("     DISPLAY:  (B)ases, (U)nits, (T)ransporters," +
                         " (C)ommodities, (D)eploy ON");
        END IF;
        CHR := ReadKey();
        IF    (CHR = "S") OR (CHR = "s")
                OUTPUT("     CONTINUING SCENARIO.");
                StopSim := FALSE;
                EXIT;
        ELSIF (CHR = "N") OR (CHR = "n")
                OUTPUT("     INPUT NEW TIME STEP (REAL Hours)");
                INPUT(TimeStep);
                OUTPUT("Time Step Set To - ", TimeStep);
        ELSIF (CHR = "B") OR (CHR = "b")
                OUTPUT("     DISPLAYING BASE MENU.");
                ASK SELF TO DisplayBaseDisplayMenu;
        ELSIF (CHR = "U") OR (CHR = "u")
                OUTPUT("     DISPLAYING UNIT MENU.");
                ASK SELF TO DisplayUnitDisplayMenu;
        ELSIF (CHR = "T") OR (CHR = "t")
                OUTPUT("     DISPLAYING TRANSPORTER MENU.");
                ASK SELF TO DisplayTransporterDisplayMenu;
        ELSIF (CHR = "C") OR (CHR = "c")
                OUTPUT("     DISPLAYING COMMODITY.");
                ASK SELF TO DisplayCommodityDisplayMenu;
        ELSIF (CHR = "D") OR (CHR = "d")
                OUTPUT("     DISPLAYING DEPLOYMENT STATUS.");
                IF Deployment
                        Deployment := FALSE;
                ELSE
                        Deployment := TRUE;
                END IF;
        ELSIF (CHR = "H") OR (CHR = "h")
```

```
                    OUTPUT("     DISPLAYING HELP MENU.");

            ELSIF (CHR = "R") OR (CHR = "r")
                    OUTPUT("   Are You Sure? (Y)");
                    CHR := ReadKey();
                    IF (CHR = "y") OR (CHR = "Y")
                            OUTPUT("     STOPPING EXECUTION.");
                            StopSim := TRUE;
                            EXIT;
                    END IF;
            ELSE
                    OUTPUT("INVALID INPUT");
            END IF;
    END LOOP;

    END METHOD;


    {-----------------------------------------------------------------------}
    ASK METHOD DisplayBaseDisplayMenu;
    {-----------------------------------------------------------------------}
    CONST

    format = "     ***************     ***************     *************** ";

    VAR

    j,n, numItems, numFOR : INTEGER;
    string : STRING;
    base1, base2, base3 : BaseObj;
    name1, name2, name3 : STRING;
    lastbase : BaseObj;
    Answer : STRING;
    CHR : CHAR;

    BEGIN

    LOOP;

            ClearScreen;
            SOUTPUT(" ",j);
            SOUTPUT(" ",j);

            lastbase := ASK Builder.OnlyBaseQ Last;
            LOOP
                    name1 := "                ";
                    name2 := "                ";
                    name3 := "                ";

                    base1 := ASK Builder.OnlyBaseQ TO Remove;
                    ASK Builder.OnlyBaseQ TO Add(base1);
                    name1 := ASK base1 Name;
                    IF base1 = lastbase
                            string := SPRINT(name1,name2,name3) WITH format;
                            SOUTPUT(string,j);
                            EXIT;
                    END IF;

                    base2 := ASK Builder.OnlyBaseQ TO Remove;
                    ASK Builder.OnlyBaseQ TO Add(base2);
                    name2 := ASK base2 Name;
                    IF base2 = lastbase
                            string := SPRINT(name1,name2,name3) WITH format;
                            SOUTPUT(string,j);
                            EXIT;
                    END IF;

                    base3:= ASK Builder.OnlyBaseQ TO Remove;
                    ASK Builder.OnlyBaseQ TO Add(base3);
                    name3 := ASK base3 Name;
                    IF base3 = lastbase
                            string := SPRINT(name1,name2,name3) WITH format;
                            SOUTPUT(string,j);
                            EXIT;
                    END IF;
                    string := SPRINT(name1,name2,name3) WITH format;
                    SOUTPUT(string,j);

            END LOOP;

            SOUTPUT(" ",j);
    SOUTPUT("=====================================================================

            SOUTPUT("     COMMAND:  (S)ingle, (A)ll, (M)odify, (R)eturn ",j);
            CHR := ReadKey();

            IF (CHR = "S") OR (CHR = "s")
                    OUTPUT("     DISPLAYING SINGLE BASE.");
                    OUTPUT("     Input Base Name then hit <RETURN>.");
                    INPUT(Answer);
                    ASK SELF TO DisplayBase(Answer);
            ELSIF (CHR = "A") OR (CHR = "a")
                    OUTPUT("     DISPLAYING ALL BASES.");
                    ASK SELF TO DisplayAllBases;
            ELSIF (CHR = "M") OR (CHR = "m")
                    OUTPUT("     MODIFYING BASE.");
                    OUTPUT("     Input Base Name then hit <RETURN>.");
                    INPUT(Answer);
                    ASK SELF TO ModifyBase(Answer);
            ELSIF (CHR = "H") OR (CHR = "h")
                    OUTPUT("     DISPLAYING HELP MENU.");

            ELSIF (CHR = "R") OR (CHR = "r")
                    OUTPUT("     RETURNING.");
                    EXIT;
            ELSE
                    OUTPUT("INVALID INPUT");
            END IF;

    END LOOP;

    END METHOD;

    {-----------------------------------------------------------------------}
    ASK METHOD DisplayBase(IN Name : STRING);
    {-----------------------------------------------------------------------}
    VAR

    Base : BaseObj;
    answer : STRING;
```

```
CHR : CHAR;

BEGIN

Base := ASK Builder.BaseQ TO FindByName(Name);
IF Base <> NILOBJ;
        LOOP
                ASK Base TO Display;
                OUTPUT("     Return? (Y)");
                CHR := ReadKey();
                IF (CHR = "Y") OR (CHR = "y");
                        ClearScreen;
                        EXIT;
                END IF;
        END LOOP;
END IF;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD DisplayAllBases;
{----------------------------------------------------------------------------}
VAR

Base: BaseObj;
i, numItems : INTEGER;
answer : STRING;
CHR : CHAR;

BEGIN

LOOP
        numItems := ASK Builder.OnlyBaseQ numberIn;
        FOR i := 1 TO numItems
                Base := ASK Builder.OnlyBaseQ TO Remove;
                ASK Builder.OnlyBaseQ TO Add(Base);
                ASK Base TO Display;
                OUTPUT("     Return? (N)");
                CHR := ReadKey();
                IF (CHR = "Y") OR (CHR = "y");
                        RETURN;
                END IF;
        END FOR;
END LOOP;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD ModifyBase(IN Name : STRING);
{----------------------------------------------------------------------------}

VAR

base : BaseObj;

BEGIN

base := ASK Builder.OnlyBaseQ TO FindByName(Name);
IF base <> NILOBJ;
        ASK base TO Modify(Builder);
```

```
END IF;

END METHOD;

{------------------------------------------------------------------------}
ASK METHOD DisplayUnitDisplayMenu;
{------------------------------------------------------------------------}
CONST

format = "     ***************     ***************     *************** ";

VAR

j ,n, numItems, numFOR : INTEGER;
string : STRING;
unit1, unit2, unit3 : UnitObj;
name1, name2, name3 : STRING;
lastunit : UnitObj;
Answer : STRING;
CHR : CHAR;
BEGIN

LOOP;

        ClearScreen;
        SOUTPUT(" ",j);
        SOUTPUT(" ",j);

        lastunit := ASK Builder.UnitQ Last;
        LOOP
                name1 := "              ";
                name2 := "              ";
                name3 := "              ";

                unit1 := ASK Builder.UnitQ TO Remove;
                ASK Builder.UnitQ TO Add(unit1);
                name1 := ASK unit1 Name;
                IF unit1 = lastunit
                        string := SPRINT(name1,name2,name3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;

                unit2 := ASK Builder.UnitQ TO Remove;
                ASK Builder.UnitQ TO Add(unit2);
                name2 := ASK unit2 Name;
                IF unit2 = lastunit
                        string := SPRINT(name1,name2,name3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;

                unit3:= ASK Builder.UnitQ TO Remove;
                ASK Builder.UnitQ TO Add(unit3);
                name3 := ASK unit3 Name;
                IF unit3 = lastunit
                        string := SPRINT(name1,name2,name3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;
                string := SPRINT(name1,name2,name3) WITH format;
                SOUTPUT(string,j);

        END LOOP;

        SOUTPUT(" ",j);
SOUTPUT("==========================================================================

        SOUTPUT("     COMMAND:  (S)ingle, Supply (O)verview, (A)ll, (M)odify, (R
        CHR := ReadKey();

        IF (CHR = "S") OR (CHR = "s")
                OUTPUT("     DISPLAYING SINGLE UNIT.");
                OUTPUT("     Input Unit Name then hit <RETURN>.");
                INPUT(Answer);
                ASK SELF TO DisplayUnit(Answer);
        ELSIF (CHR = "O") OR (CHR = "o")
                OUTPUT("      DISPLAYING UNIT SUPPLY OVERVIEW");
                OUTPUT("     Input Unit Name then hit <RETURN>.");
                INPUT(Answer);
                unit1 := ASK Builder.UnitQ TO FindByName(Answer);
                IF unit1 <> NILOBJ
                        ASK SELF TO DisplayBaseOverView(unit1);
                        OUTPUT("     Press any key to continue.");
                        CHR := ReadKey();
                END IF;
        ELSIF (CHR = "A") OR (CHR = "a")
                OUTPUT("     DISPLAYING ALL UNITS.");
                ASK SELF TO DisplayAllUnits;
        ELSIF (CHR = "M") OR (CHR = "m")
                OUTPUT("     MODIFYING UNIT.");
                OUTPUT("     Input Unit Name then hit <RETURN>.");
                INPUT(Answer);
                ASK SELF TO ModifyUnit(Answer);
        ELSIF (CHR = "H") OR (CHR = "h")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r")
                OUTPUT("     RETURNING.");
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

END LOOP;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD DisplayUnit(IN Name : STRING);
{----------------------------------------------------------------------------}

VAR

Unit : UnitObj;
answer : STRING;
CHR : CHAR;

BEGIN
```

```
Unit := ASK Builder.UnitQ TO FindByName(Name);
IF Unit <> NILOBJ;
        LOOP
                ASK Unit TO Display;
                OUTPUT("      Return? (Y)");
                CHR := ReadKey();
                IF (CHR <> "N") OR (CHR <> "N");
                        ClearScreen;
                        EXIT;
                END IF;
        END LOOP;
END IF;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD DisplayAllUnits;
{----------------------------------------------------------------------------}
VAR

Unit: UnitObj;
i, numItems : INTEGER;
answer : STRING;
CHR : CHAR;


BEGIN

LOOP
        numItems := ASK Builder.UnitQ numberIn;
        FOR i := 1 TO numItems
                Unit := ASK Builder.UnitQ TO Remove;
                ASK Builder.UnitQ TO Add(Unit);
                ASK Unit TO Display;
                OUTPUT("      Return? (N)");
                CHR := ReadKey();
                IF (CHR = "y") OR (CHR = "Y")
                        ClearScreen;
                        RETURN;
                END IF;
        END FOR;
END LOOP;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD ModifyUnit(IN Name : STRING);
{----------------------------------------------------------------------------}

VAR

unit : UnitObj;

BEGIN

unit := ASK Builder.UnitQ TO FindByName(Name);
IF unit <> NILOBJ;
        ASK unit TO Modify(Builder);
END IF;
END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD DisplayTransporterDisplayMenu;
{-----------------------------------------------------------------------}
CONST

format = "     ***************     ***************     *************** ";

VAR

string, string1, string2, string3 : STRING;
transporter, lasttransporter : TransporterObj;
name : STRING;
Answer, Name : STRING;
CHR : CHAR;
j ,ID, VehicleID : INTEGER;

BEGIN

LOOP;

        ClearScreen;
        SOUTPUT(" ",j);
        SOUTPUT(" ",j);

        lasttransporter := ASK Builder.BigTransporterQ Last;
        LOOP
                string1 := "              ";
                string2 := "              ";
                string3 := "              ";

                transporter := ASK Builder.BigTransporterQ TO Remove;
                ASK Builder.BigTransporterQ TO Add(transporter);
                name := ASK transporter Name;
                ID := ASK transporter VehicleID;
                string1 := name + "-" + INTTOSTR(ID);
                IF transporter = lasttransporter
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;

                transporter := ASK Builder.BigTransporterQ TO Remove;
                ASK Builder.BigTransporterQ TO Add(transporter);
                name := ASK transporter Name;
                ID := ASK transporter VehicleID;
                string2 := name + "-" + INTTOSTR(ID) ;
                IF transporter = lasttransporter
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;

                transporter := ASK Builder.BigTransporterQ TO Remove;
                ASK Builder.BigTransporterQ TO Add(transporter );
                name  := ASK transporter  Name;
                ID  := ASK transporter  VehicleID;
                string3 := name + "-" + INTTOSTR(ID) ;
                IF transporter  = lasttransporter
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;
                string := SPRINT(string1,string2,string3) WITH format;
                SOUTPUT(string,j);

        END LOOP;

        SOUTPUT(" ",j);
SOUTPUT("================================================================

        SOUTPUT("     COMMAND:  (S)ingle, (P)repo, (A)ll, (M)odify, (R)eturn ",j
        CHR := ReadKey();

        IF (CHR = "S") OR (CHR = "s")
                OUTPUT("     DISPLAYING SINGLE TRANSPORTER.");
                OUTPUT("     Input Transporter Model then hit <RETURN>.");
                INPUT(Name);
                OUTPUT("     Input Transporter ID Number.");
                INPUT(VehicleID);
                ASK SELF TO DisplayTransporter(Name,VehicleID);
        ELSIF (CHR = "P") OR (CHR = "p")
                ASK SELF TO DisplayPrepoMenu;
        ELSIF (CHR = "A") OR (CHR = "a")
                OUTPUT("     DISPLAYING ALL TRANSPORTERS.");
                ASK SELF TO DisplayAllTransporters;
        ELSIF (CHR = "M") OR (CHR = "m")
                OUTPUT("     MODIFYING TRANSPORTER.");
                ASK SELF TO ModifyTransporter;
        ELSIF (CHR = "H") OR (CHR = "h")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r")
                OUTPUT("     RETURNING.");
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

END LOOP;

END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD DisplayTransporter(IN Name : STRING; IN VehicleID : INTEGER);
{--------------------------------------------------------------------------}
VAR
Transporter : TransporterObj;
i, numItems : INTEGER;
answer : STRING;
CHR : CHAR;

BEGIN

numItems := ASK Builder.BigTransporterQ numberIn;
FOR i := 1 TO numItems
        Transporter := ASK Builder.BigTransporterQ TO Remove;
        ASK Builder.BigTransporterQ TO Add(Transporter);
        IF (Transporter.Name = Name) AND (Transporter.VehicleID = VehicleID)
                LOOP
                        ASK Transporter TO Display;
                        OUTPUT("     Return? (Y or N)");
                        CHR := ReadKey();
                        IF (CHR = "y") OR (CHR = "Y")
                                ClearScreen;
                                EXIT;
                        END IF;
                END LOOP;
        END IF;
END FOR;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD DisplayPrepoMenu;
{---------------------------------------------------------------------------}
VAR

transporter : TransporterObj;
answer : STRING;
destination : BaseObj;
shipment : ShipmentObj;
j : INTEGER;
CHR : CHAR;


BEGIN

LOOP
OUTPUT(" ");
OUTPUT("      (A)ctivate prepositioned Transporter, (R)eturn");
CHR := ReadKey();
IF (CHR = "A") OR (CHR = "a")
        ClearScreen;
        j := 0;
        ASK Builder.PrepoTransporterQ TO Display(j);
        OUTPUT("");
        OUTPUT("     Input prepositioned Transporter Name");
        INPUT(answer);
        transporter := ASK Builder.PrepoTransporterQ TO FindByName(answer);
        IF transporter <> NILOBJ
                shipment := ASK transporter.Cargo First;
                destination := ASK shipment.Route First;
                TELL transporter TO GoTo(destination);
                OUTPUT(transporter.Name, " activated.  Proceeding to ", destinat
        END IF;
ELSIF (CHR = "R") OR (CHR = "r")
        EXIT;
END IF;




END LOOP;

END METHOD;
```

```
{--------------------------------------------------------------------------}
ASK METHOD DisplayAllTransporters;
{--------------------------------------------------------------------------}
VAR

Transporter : TransporterObj;
i, numItems : INTEGER;
answer : STRING;
CHR : CHAR;

BEGIN

LOOP
        numItems := ASK Builder.BigTransporterQ numberIn;
        FOR i := 1 TO numItems
                Transporter := ASK Builder.BigTransporterQ TO Remove;
                ASK Builder.BigTransporterQ TO Add(Transporter);
                ASK Transporter TO Display;
                OUTPUT("     Return? (Y or N)");
                CHR := ReadKey();
                IF (CHR = "Y") OR (CHR = "y")
                        ClearScreen;
                        RETURN;
                END IF;

        END FOR;
END LOOP;

END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD ModifyTransporter;
{--------------------------------------------------------------------------}
VAR

transporter, newtransporter : TransporterObj;
answer : STRING;
base : BaseObj;
port : PortObj;
i, numItems, integer : INTEGER;
CHR : CHAR;


BEGIN

LOOP
OUTPUT(" ");
OUTPUT("     (A)dd Transporter, (D)estroy Transporter, (R)eturn");
CHR := ReadKey();
IF (CHR = "A") OR (CHR = "a")

        OUTPUT("     Input Transporter Model Name");
        INPUT(answer);
        transporter := ASK Builder.TransporterQ TO FindByName(answer);
        IF transporter <> NILOBJ
                ASK transporter TO SetVehicleID(transporter.VehicleID + 1);
                newtransporter := CLONE(transporter);
                OUTPUT("     Input Base or Unit where transporter will begin.");
                INPUT(answer);
                base := ASK Builder.BaseQ TO FindByName(answer);
                IF base <> NILOBJ
                        CASE transporter.Class
                                WHEN Aircraft :
                                        port := base.AirPort;
                                WHEN Ship :
                                        port := base.SeaPort;
                                WHEN Rail :
                                        port := base.RailYard;
                                WHEN Truck :
                                        port := base.TruckStop;
                        END CASE;
                        ASK port TO GetArrival(newtransporter);
                        ASK Builder.BigTransporterQ TO Add(newtransporter);
                        ASK newtransporter TO SetPort(base);
                        ASK newtransporter TO SetLocation(base);
                        ASK newtransporter TO SetPosition(base.Position);
                END IF;
        END IF;

ELSIF (CHR = "D") OR (CHR = "d")

        OUTPUT("     Are you sure? (Y) ");
        CHR := ReadKey();
        IF (CHR = "Y") OR (CHR = "y")
                OUTPUT("     Input Transporter Model Name");
                INPUT(answer);
                OUTPUT("     Input Transporter ID");
                INPUT(integer);
                numItems := ASK Builder.BigTransporterQ numberIn;
                FOR i := 1 TO numItems
                        transporter := ASK Builder.BigTransporterQ TO Remove;
                        ASK Builder.BigTransporterQ TO Add(transporter);
                        IF (transporter.Name = answer) AND
                                        (transporter.VehicleID = integer)
                                ASK transporter TO CleanUp;
                        END IF;
                END FOR;
        END IF;

ELSIF (CHR = "R") OR (CHR = "r")
        EXIT;
END IF;



END LOOP;

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD DisplayCommodityDisplayMenu;
{----------------------------------------------------------------------}
CONST
format = "     **************** ********* **** x *** x *** ******** ********** **

VAR
j,i, numItems : INTEGER;
string : STRING;
```

```
Commodity, lastcommodity : CommodityObj;
name : STRING;
Answer : STRING;
CHR : CHAR;
ID : INTEGER;

BEGIN

LOOP

j := 1;
ASK Builder.CommodityQ TO Display(j);

        SOUTPUT("     COMMAND:  (L)ocate, (H)elp, (R)eturn ", j);
        CHR := ReadKey();

        IF (CHR = "L") OR (CHR = "l")
                OUTPUT("     LOCATING COMMODITY.");
                OUTPUT("     Input Commodity Name then hit <RETURN>.");
                INPUT(Answer);
                ASK SELF TO DisplayCommodity(Answer);
        ELSIF (CHR = "H") OR (CHR = "h")
                OUTPUT("     DISPLAYING HELP MENU.");

        ELSIF (CHR = "R") OR (CHR = "r")
                OUTPUT("     RETURNING.");
                EXIT;
        ELSE
                OUTPUT("INVALID INPUT");
        END IF;

END LOOP;

END METHOD;

{-------------------------------------------------------------------------}
ASK METHOD DisplayCommodity(IN Name : STRING);
{-------------------------------------------------------------------------}


{Displays all commodities in their locations}

CONST
format = "      *************** ********  ********      ********      ******** ";
format2 = "      ***************  *** ********";

VAR

i,j, numItems, numItems2 : INTEGER;
base : BaseObj;
unit : UnitObj;
transporter : TransporterObj;
shipment : ShipmentObj;
commodity : CommodityObj;
rate : REAL;
string : STRING;
answer : STRING;
CHR : CHAR;

BEGIN
```

```
LOOP

ClearScreen;
SOUTPUT(" ",j);
SOUTPUT(  "     Locating "+ Name+ ":  BASES",j);
SOUTPUT(" ",j);
SOUTPUT(  "     Base Name        On Hand  On Order Stocking Obj  Consumption",j)
SOUTPUT("================================================================

numItems := ASK Builder.OnlyBaseQ numberIn;
FOR i := 1 TO numItems
        base := ASK Builder.OnlyBaseQ TO Remove;
        ASK Builder.OnlyBaseQ TO Add(base);
        commodity := ASK base.Inventory TO FindByName(Name);
        IF commodity <> NILOBJ
                rate := 0.00;
                string := SPRINT(base.Name, commodity.OnHand, commodity.OnOrder,
                                         commodity.StockTo, rate) WITH format;
                SOUTPUT(string,j);
        END IF;
END FOR;
SOUTPUT("================================================================
SOUTPUT(" ",j);
SOUTPUT("    Continue? (Y)",j);
CHR := ReadKey();
IF (CHR = "N") OR (CHR = "n")
        EXIT;
END IF;

ClearScreen;
j := 0;
SOUTPUT(" ",j);
SOUTPUT(  "     Locating "+ Name+ ":  UNITS",j);
SOUTPUT(" ",j);
SOUTPUT(  "     Unit Name        On Hand  On Order Stocking Obj  Consumption",j)
SOUTPUT("================================================================

numItems := ASK Builder.UnitQ numberIn;
FOR i := 1 TO numItems
        unit := ASK Builder.UnitQ TO Remove;
        ASK Builder.UnitQ TO Add(unit);
        commodity := ASK unit.Inventory TO FindByName(Name);
        IF commodity <> NILOBJ
                CASE unit.CombatIntensity
                        WHEN High :
                                rate := commodity.HighRate;
                        WHEN Med :
                                rate := commodity.MedRate;
                        WHEN Low :
                                rate := commodity.LowRate;
                        WHEN None :
                                rate := 0.00;
                        OTHERWISE
                                rate := 0.00;
                END CASE;
                string := SPRINT(unit.Name, commodity.OnHand, commodity.OnOrder,
                                         commodity.StockTo, rate) WITH format;
                SOUTPUT(string,j);
        END IF;
END FOR;
SOUTPUT("=======================================================================
SOUTPUT(" ",j);
SOUTPUT("     Continue? (Y)",j);
CHR := ReadKey();
IF (CHR = "N") OR (CHR = "n")
        EXIT;
END IF;


ClearScreen;
SOUTPUT(" ",j);
SOUTPUT(  "      Locating "+ Name+ ":  TRANSPORTERS",j);
SOUTPUT(" ",j);
SOUTPUT(  "      Name              ID   On Hand",j);
SOUTPUT("=======================================================================

numItems := ASK Builder.BigTransporterQ numberIn;
FOR i := 1 TO numItems
        transporter := ASK Builder.BigTransporterQ TO Remove;
        ASK Builder.BigTransporterQ TO Add(transporter);
        numItems2 := ASK transporter.Cargo numberIn;
        FOR j := 1 TO numItems2

                shipment := ASK transporter.Cargo TO Remove;
                ASK transporter.Cargo TO Add(shipment);
                commodity := ASK shipment Item;
                IF commodity.Name = Name
                        string := SPRINT(transporter.Name,
                          transporter.VehicleID, commodity.OnHand) WITH format2;
                        SOUTPUT(string,j);
                END IF;
        END FOR;
END FOR;
SOUTPUT("=======================================================================
SOUTPUT(" ",j);
SOUTPUT("     Return? (Y)",j);
CHR := ReadKey();

IF (CHR <> "N") AND (CHR <> "n")
        ClearScreen;
        EXIT;
END IF;

END LOOP;

END METHOD;


{-----------------------------------------------------------------------------}
ASK METHOD DisplayOverView(IN BaseQ : BaseQObj);
{-----------------------------------------------------------------------------}
VAR

Stats : StatObj;

BEGIN

NEW(Stats);
```

```
ASK Stats TO CollectAllData(BaseQ);
ASK Stats TO DisplayOverView;

DISPOSE(Stats);

END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD DisplayDeploymentOverView(IN BaseQ : BaseQObj);
{--------------------------------------------------------------------------}
VAR

Stats : StatObj;

BEGIN

NEW(Stats);

ASK Stats TO CollectAllDeploymentData(BaseQ);
ASK Stats TO DisplayOverView;

DISPOSE(Stats);

END METHOD;


{--------------------------------------------------------------------------}
ASK METHOD DisplayBaseDeploymentOverView(IN Base : BaseObj);
{--------------------------------------------------------------------------}
VAR

Stats : StatObj;

BEGIN

NEW(Stats);

ASK Stats TO CollectDeploymentData(Base);
ASK Stats TO DisplayOverView;

DISPOSE(Stats);

END METHOD;


{--------------------------------------------------------------------------}
ASK METHOD DisplayBaseOverView(IN Base : BaseObj);
{--------------------------------------------------------------------------}
VAR

Stats : StatObj;

BEGIN

NEW(Stats);

ASK Stats TO CollectData(Base);
ClearScreen;
ASK Stats TO DisplayOverView;
```

```
DISPOSE(Stats);

END METHOD;

{------------------------------------------------------------------------}
ASK METHOD DisplayHelpMenu;
{------------------------------------------------------------------------}
VAR

Answer : STRING;

BEGIN

END METHOD;

{------------------------------------------------------------------------}
ASK METHOD ReadScenarioMasterFile;
{------------------------------------------------------------------------}

CONST

MasterFile = "Scenes.mst";

VAR

File : StreamObj;
object : NamedObj;
string : STRING;
i, integer : INTEGER;

BEGIN

NEW(File);
ASK File TO Open(MasterFile, Input);


ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i := 1 TO integer

        ASK File TO ReadString(string);
        NEW(object);
        ASK object TO SetName(string);
        ASK ScenarioList TO Add(object);
        ASK File TO ReadLine(string);
END FOR;
ASK File TO Close;
DISPOSE(File);

END METHOD;

END OBJECT;

END MODULE.
```

```
DEFINITION MODULE DeBug;

{Comments}

{Type Declarations}

TYPE

PROCEDURE DebugBreak();

END MODULE.
```

```
IMPLEMENTATION MODULE DeBug;

{Comments}

{Import statements}

FROM SimMod IMPORT SimTime;

{Definitions}

{-----------------------------------------------------------------------}
PROCEDURE DebugBreak();
{-----------------------------------------------------------------------}

{EMPTY PROCEDURE TO ALLOW DEBUGGER TO CALL BREAK.  TO UTILIZE, HAVE PROGRAM
CALL THIS PROCEDURE WHERE YOU WANT A DEBUG BREAK.  MAKE SURE YOU SET THE BREAK
WHEN YOU BEGIN THE -msdebug RUN.  SEE DEBNUG MANUAL FOR DEBUGGING INSTRUCTIONS}

VAR

BEGIN

OUTPUT(SimTime);

END PROCEDURE;


END MODULE.
```

```
DEFINITION MODULE Distant;

{Distant provides all the procedures for manipulating latitudes and longtiude
into distances.  also contains the PositionRecType definition}

TYPE

MinType = [0..59];

LatDegType = [0..90];
LatDirType = STRING;

LongDegType = [0..180];
LongDirType = STRING;


PositionRecType = RECORD

        LatDeg : LatDegType;
        LatMin : MinType;
        LatDir : LatDirType;
        LongDeg : LongDegType;
        LongMin : MinType;
        LongDir : LongDirType;

END RECORD;


PROCEDURE LatTOphi(IN Deg : LatDegType; IN Min : MinType;
                                                IN Dir : LatDirType) : REAL;

PROCEDURE LongTOtheta(IN Deg : LongDegType; IN Min : MinType;
                                                IN Dir : LongDirType) : REAL;

PROCEDURE CalcDistance(IN Loc1, Loc2 : PositionRecType) : REAL;

PROCEDURE InputPosition() : PositionRecType;
PROCEDURE OutputPosition(IN Position : PositionRecType; INOUT j : INTEGER);

END MODULE;
```

```
IMPLEMENTATION MODULE Distant;

FROM MathMod IMPORT POWER,
                    SQRT,
                    COS,
                    SIN,
                    ASIN,
                    pi;
FROM SOUTPUT IMPORT SOUTPUT;

{------------------------------------------------------------------------}
PROCEDURE LatTOphi(IN Deg : LatDegType; IN Min : MinType;
                                          IN Dir : LatDirType) : REAL;
{------------------------------------------------------------------------}

{CONVERTS LATITUDE TO PHI ANGLE IN SPHERICAL COORDINATES}

VAR

phi : REAL;

BEGIN

CASE Dir

        WHEN 'N' :
                phi := (90.00 - (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0)));
        WHEN 'n' :
                phi := (90.00 - (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0)));

        WHEN 'S' :
                phi := (90.00 + (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0)));
        WHEN 's' :
                phi := (90.00 + (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0)));
        OTHERWISE
                OUTPUT("Lat direction out of range - LatTOphi");
                HALT;
END CASE;
IF (phi < 0.0)
        phi := 0.0;
END IF;
IF (phi > 180.0)
        phi := 180.0;
END IF;
phi := (phi/360.0) * 2.0 * pi;

RETURN(phi);

END PROCEDURE;


{------------------------------------------------------------------------}
PROCEDURE LongTOtheta(IN Deg : LongDegType; IN Min : MinType;
                                          IN Dir : LongDirType) : REAL;
{------------------------------------------------------------------------}

{CONVERTS LONGITUDE TO THETA ANGLE IN SPHERICAL COORDINATES}

VAR
```

```
theta : REAL;

BEGIN

CASE Dir

        WHEN 'E' :
                theta := (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0));
        WHEN 'e' :
                theta := (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0));

        WHEN 'W' :
                theta := (360.00 - (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0)));
        WHEN 'w' :
                theta := (360.00 - (FLOAT(Deg) + ((60.0 - FLOAT(Min))/60.0)));
        OTHERWISE
                OUTPUT("Long Direction out of Range");
                HALT;
END CASE;
IF (theta < 0.0)
        theta := 0.0;
END IF;
IF (theta > 360.0)
        theta := 360.0;
END IF;
theta := (theta/360.0) * 2.0 * pi;

RETURN(theta);

END PROCEDURE;


{---------------------------------------------------------------------------}
PROCEDURE CalcDistance(IN Node1, Node2 : PositionRecType) : REAL;
{---------------------------------------------------------------------------}

{CALCULATES DISTANCE BETWEEN TO LAT LONG POSITIONS}

CONST
rho = 3441.251855; {Earth Diameter
                        equatorial 7926 sm, 12755 km, 6882.50371 nm
                        polar 7900 sm, 12714 km 6860.30413 nm
                     1 km = .62137sm
                     1 nm = 6080.20 ft
                          = 1.85325 km}

VAR
phi1, phi2 : REAL;
theta1, theta2 : REAL;
x1,x2 : REAL;
y1,y2 : REAL;
z1,z2 : REAL;
StraightDistance : REAL;
ArcAngle : REAL;
ArcDistance : REAL;

BEGIN

phi1 := LatTOphi(Node1.LatDeg, Node1.LatMin, Node1.LatDir);
```

```
phi2 := LatTOphi(Node2.LatDeg, Node2.LatMin, Node2.LatDir);
theta1 := LongTOtheta(Node1.LongDeg, Node1.LongMin, Node1.LongDir);
theta2 := LongTOtheta(Node2.LongDeg, Node2.LongMin, Node2.LongDir);

x1 := rho * (SIN(phi1)) * COS(theta1);
x2 := rho * (SIN(phi2)) * COS(theta2);
y1 := rho * (SIN(phi1)) * SIN(theta1);
y2 := rho * (SIN(phi2)) * SIN(theta2);
z1 := rho * COS(phi1);
z2 := rho * COS(phi2);

StraightDistance := SQRT(POWER((x1 - x2), 2.0) + POWER((y1 - y2), 2.0) +
                                             POWER((z1 - z2), 2.0));

ArcAngle := 2.0 * ASIN((StraightDistance/2.0)/rho);
ArcDistance := rho * ArcAngle;
RETURN(ArcDistance);

END PROCEDURE;

{------------------------------------------------------------------------}
PROCEDURE InputPosition() : PositionRecType;
{------------------------------------------------------------------------}

{ALLOW USER TO INTERACTIVELY INPUT A POSITION WHEN CREATING AN OBJECT IN.
CALLED BY SCENARIO EDITOR AND MODIFY METHODS OF BASES AND UNITS}

VAR

position : PositionRecType;
latdeg : LatDegType;
latdir : LatDirType;
longdeg : LongDegType;
longdir : LongDirType;
min : MinType;
ok : BOOLEAN;
BEGIN

NEW(position);


LOOP
        ok := TRUE;
        OUTPUT("     Input Latitude (DD MM H)");
        INPUT(latdeg, min, latdir);

        IF (latdeg >= 0) AND (latdeg <= 90)
                position.LatDeg := latdeg;
        ELSE ok := FALSE
        END IF;

        IF (min >= 0) AND (min <= 59)
                position.LatMin := min;
        ELSE ok := FALSE
        END IF;

        IF (latdir = "n") OR (latdir = "N") OR (latdir = "S") OR (latdir = "s")
                position.LatDir := latdir;
        ELSE ok := FALSE
        END IF;
```

```
          IF ok = TRUE
                 EXIT
          END IF;
    END LOOP;

    LOOP
          ok := TRUE;
          OUTPUT("      Input Longitude (DDD MM H)");
          INPUT(longdeg, min, longdir);

          IF (longdeg >= 0) AND (longdeg <= 180)
                 position.LongDeg := longdeg;
          ELSE ok := FALSE
          END IF;

          IF (min >= 0) AND (min <= 59)
                 position.LongMin := min;
          ELSE ok := FALSE
          END IF;

          IF (longdir = "E") OR (longdir = "e") OR (longdir = "W") OR (longdir = "
                 position.LongDir := longdir;
          ELSE ok := FALSE
          END IF;

          IF ok = TRUE
                 EXIT
          END IF;
    END LOOP;

    RETURN(position);

    END PROCEDURE;

    {----------------------------------------------------------------------}
    PROCEDURE OutputPosition(IN Position : PositionRecType; INOUT j : INTEGER);
    {----------------------------------------------------------------------}

    {OUTPUTS POSITION TO THE SCREEN}

    CONST

    format = "     **********  *** ** *";
    VAR

    string : STRING;

    BEGIN

    IF Position <> NILREC
           OUTPUT("Position:  ");
           string := SPRINT("Latitude:", Position.LatDeg, Position.LatMin,
                  Position.LatDir) WITH format;
           OUTPUT(string);
           string := SPRINT("Longitude:", Position.LongDeg, Position.LongMin,
                  Position.LongDir) WITH format;
           OUTPUT(string);
    j := j + 3;
    END IF;
```

```
END PROCEDURE;
END MODULE;
```

```
DEFINITION MODULE LogMan;

{Logman is an administrative entity which routes Shipment orders to the
proper supplying BaseObj.  also build the best route to the recipient}

{Import statements}

FROM Base IMPORT BaseObj,
                 BaseQObj,
                 ALL BaseGroupType;
FROM CommodQ IMPORT CommodityObj;

FROM Unit IMPORT UnitObj;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;
FROM Supply IMPORT Supply;
{
FROM IMPORT
}

{Type Declarations}

TYPE


{==========================================================================}
LogisticsManagerObj  = OBJECT
{==========================================================================}

{FIELDS}

ConusQ : BaseQObj;
ILocQ : BaseQObj;
TheaterQ : BaseQObj;
UnitQ : BaseQObj;
POEQ : BaseQObj;
PODQ : BaseQObj;
POSQ : BaseQObj;
DeadShipmentsQ : ShipmentQObj;
{METHODS}
{} ASK METHOD HandleUnitRequest(IN Requester : UnitObj; INOUT Item :
                                                               CommodityObj);
{} ASK METHOD HandleBaseRequest(IN Requester : BaseObj;  INOUT Item :
                                                               CommodityObj);
{x} ASK METHOD PickSupplier(IN Requester : BaseObj; IN Item : CommodityObj) :
                                                                 BaseObj;

{x} ASK METHOD PickBestSupplierInGroup(IN Requester : BaseObj; IN Item :
                                CommodityObj; IN Group : BaseQObj) : BaseObj;
{x} ASK METHOD PickBestBase(IN Base1: BaseObj; IN Group : BaseQObj; INOUT
                                               Mode : INTEGER) : BaseObj;
{x} ASK METHOD PickClosestBaseInGroup(IN Base1: BaseObj; IN Group : BaseQObj) :
                                                                       BaseObj;
{x} ASK METHOD BuildShipment(IN Requester : BaseObj; IN Item : CommodityObj; IN
                                              Route : BaseQObj) : ShipmentObj;
{x} ASK METHOD BuildRoute(IN Origin : BaseObj; IN Destination : BaseObj; IN Item
                               : CommodityObj; IN Surge : BOOLEAN) : BaseQObj;
{
}
{x} ASK METHOD PickBestILoc(IN ConusBase : BaseObj; IN TheaterBase : BaseObj) :
                                                               BaseObj;
```

```
}
ASK METHOD ObjInit;
ASK METHOD ObjTerminate;

END OBJECT;

PROCEDURE Communicates(IN Base1, Base2 : BaseObj) : BOOLEAN;

VAR

LogisticsManager : LogisticsManagerObj;

END MODULE.
```

```
IMPLEMENTATION MODULE LogMan;

{Comments}

{Import statements}

FROM Base IMPORT BaseObj,
                 BaseQObj,
                 ALL BaseGroupType;
FROM CommodQ IMPORT CommodityObj;
FROM Shpmnt IMPORT ShipmentObj;
FROM Unit IMPORT UnitObj;
FROM Distant IMPORT CalcDistance;
FROM Supply IMPORT Supply;
{
FROM IMPORT ;
}

{Definitions}

{=====================================================================}
OBJECT LogisticsManagerObj;
{=====================================================================}

     {METHODS}
{---------------------------------------------------------------------}
ASK METHOD HandleUnitRequest(IN Requester : UnitObj; INOUT Item : CommodityObj);
{---------------------------------------------------------------------}

{HANDLES UNIT REQUEST FOR COMMODITY}


VAR

OrderQty, Difference : REAL;
Origin : BaseObj;
OriginItem : CommodityObj;
Shipment : ShipmentObj;
Route : BaseQObj;

BEGIN

OrderQty := Item.StockTo - (Item.OnOrder + Item.OnHand);

{CHECK IF COMMODITY DEPLOYMENT AND UNIT IS DEPLOYING}

IF (ASK Item Deployment) AND (NOT(ASK Requester InPlace))

{IF SO, SUPPLIER IS UNIT ORIGIN ROUTE ACCORDINGLY}

        Origin := ASK Requester Origin;
        Route := ASK SELF TO BuildRoute(Origin, Requester, Item, TRUE);
        Shipment := ASK SELF TO BuildShipment(Requester,Item,Route);
        ASK Origin TO FillOrder(Shipment);
        ASK Item TO AddOnOrder(OrderQty);

ELSE

{IF NOT, SUPPLIER IS CLOSEST BASE THAT STOCK THE COMMODITY, LOOK FOR ELIGIBLE
SUPPLIERS UNTIL ORDER IS FILLED IE. ALLOW PARTIAL FILLS}
```

```
        WHILE OrderQty >= 1.0
                Origin := ASK SELF TO PickSupplier(Requester, Item);
                Route := ASK SELF TO BuildRoute(Origin,Requester, Item, FALSE);
                Shipment := ASK SELF TO BuildShipment(Requester,Item,Route);
                OriginItem := ASK Origin.Inventory TO FindByName(Item.Name);

                IF OriginItem <> NILOBJ
                IF OriginItem.OnHand >= OrderQty

                        ASK Origin TO FillOrder(Shipment);
                        ASK Item TO AddOnOrder(OrderQty);

                        OrderQty := 0.0;
                ELSIF (OriginItem.OnHand < 1.0)
                        ASK Origin TO FillOrder(Shipment);
                        ASK Item TO AddOnOrder(OrderQty);
                        OrderQty := 0.0;
                ELSE
                        Difference := OrderQty - OriginItem.OnHand;
                        ASK Shipment.Item TO SetOnHand(0.0);
                        ASK Shipment.Item TO SetOnOrder(0.0);
                        ASK Shipment.Item TO SetStockTo(OriginItem.OnHand);

                        ASK Origin TO FillOrder(Shipment);
                        ASK Item TO AddOnOrder(Shipment.Item.StockTo);

                        OrderQty := Difference;
                END IF;
                END IF;
        END WHILE;
END IF;

END METHOD;


{----------------------------------------------------------------------------}
ASK METHOD HandleBaseRequest(IN Requester : BaseObj; INOUT Item : CommodityObj);
{----------------------------------------------------------------------------}

{HANDLES BASE REQEST FOR COMMODITY}

VAR

OrderQty, Difference : REAL;
Origin : BaseObj;
OriginItem, NewItem : CommodityObj;
Shipment : ShipmentObj;
Route : BaseQObj;


BEGIN

{ALWAYS PICK CLOSEST ELIGIBLE BASE, UNLIKE UNIT CASE}

OrderQty := Item.StockTo - (Item.OnOrder + Item.OnHand);
        WHILE OrderQty >= 1.0
                Origin := ASK SELF TO PickSupplier(Requester, Item);
                Route := ASK SELF TO BuildRoute(Origin,Requester, Item, FALSE);
                Shipment := ASK SELF TO BuildShipment(Requester,Item,Route);
                OriginItem := ASK Origin.Inventory TO FindByName(Item.Name);


                IF OriginItem <> NILOBJ
                        IF (OriginItem.OnHand >= OrderQty) OR
                        (Origin.Name = "Supply")
                                ASK Origin TO FillOrder(Shipment);
                                ASK Item TO AddOnOrder(OrderQty);
                                OrderQty := 0.0;
                        ELSIF (OriginItem.OnHand < 1.0)
                                ASK Origin TO FillOrder(Shipment);
                                ASK Item TO AddOnOrder(OrderQty);
                                OrderQty := 0.0;
                        ELSE
                                Difference := OrderQty - OriginItem.OnHand;
                                ASK Shipment.Item TO SetOnHand(0.0);
                                ASK Shipment.Item TO SetOnOrder(0.0);
                                ASK Shipment.Item TO
                                         SetStockTo(OriginItem.OnHand);

                                ASK Item TO AddOnOrder(OrderQty);
                                ASK Origin TO FillOrder(Shipment);

                                OrderQty := Difference;
                        END IF;
                ELSE
                        {
                        OUTPUT("Backordering commodity not in inventory");
                        }
                        NewItem := CLONE(Item);
                        ASK Origin.Inventory TO Add(NewItem);
                        ASK NewItem TO SetOnHand(0.0);
                        ASK NewItem TO SetOnOrder(0.0);
                        ASK NewItem TO SetStockTo(OrderQty);
                        ASK NewItem TO SetOrderAt(0.0);
                        ASK NewItem TO SetEmerOrderAt(0.0);

                        ASK Origin TO BackOrderStuff(Shipment);
                        ASK Item TO AddOnOrder(OrderQty);
                        OrderQty := 0.0;
                END IF;
        END WHILE;

END METHOD;

{--------------------------------------------------------------------}
ASK METHOD PickSupplier(IN Requester : BaseObj; IN Item : CommodityObj) :
                                                                 BaseObj;
{--------------------------------------------------------------------}

{PICKS BEST SUPPLIER BASED ON REQUESTER GROUP}

VAR
i, numItems : INTEGER;
Supplier : BaseObj;
CurrentBase : BaseObj;
CurrentItem : CommodityObj;
BestDistance : REAL;
CurrentGroup : BaseQObj;
```

```
BEGIN
Supplier := NILOBJ;
CASE Requester.Group

WHEN UNIT:
        Supplier := PickBestSupplierInGroup(Requester,Item,TheaterQ);
        IF Supplier = NILOBJ
                Supplier := PickBestSupplierInGroup(Requester,Item,ILocQ);
        END IF;
        IF Supplier = NILOBJ
                Supplier := PickBestSupplierInGroup(Requester,Item,ConusQ);
        END IF;
        IF Supplier = NILOBJ
                Supplier := PickClosestBaseInGroup(Requester,ConusQ);
        END IF;

WHEN THEATER:
        Supplier := PickBestSupplierInGroup(Requester,Item,ILocQ);
        IF Supplier = NILOBJ
                Supplier := PickBestSupplierInGroup(Requester,Item,ConusQ);
        END IF;
        IF Supplier = NILOBJ
                Supplier := PickClosestBaseInGroup(Requester,ConusQ);
        END IF;

WHEN ILOC:
        Supplier := PickBestSupplierInGroup(Requester,Item,ConusQ);
        IF Supplier = NILOBJ
                Supplier := PickClosestBaseInGroup(Requester,ConusQ);
        END IF;

WHEN CONUS:
        Supplier := ASK ConusQ TO FindByName("Supply");

OTHERWISE
        Supplier := PickClosestBaseInGroup(Requester,ConusQ);

END CASE;

RETURN(Supplier);
END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD PickBestSupplierInGroup(IN Requester : BaseObj; IN Item :
                                  CommodityObj; IN Group : BaseQObj) : BaseObj;  {

{RETURNS CLOSEST BASE THAT STOCKS THE DESIRED COMMODITY, IF NO BASE, RETURNS
NILOBJ}


VAR

i, numItems : INTEGER;
Supplier : BaseObj;
CurrentBase : BaseObj;
CurrentItem : CommodityObj;
BestDistance, NewDistance : REAL;

BEGIN
Supplier := NILOBJ;
```

```
numItems := ASK Group numberIn;
FOR i := 1 TO numItems
        CurrentBase := ASK Group TO Remove;
        ASK Group TO Add(CurrentBase);
        IF CurrentBase.Name <> "Supply"
        CurrentItem := ASK CurrentBase.Inventory TO
                                                FindByName(Item.Name);
        IF CurrentItem <> NILOBJ
        IF CurrentItem.OnHand >= 1.0
                NewDistance :=
                        CalcDistance(Requester.Position,CurrentBase.Position);

                IF Supplier = NILOBJ
                        Supplier := CurrentBase;
                        BestDistance := NewDistance;
                ELSIF NewDistance < BestDistance
                        BestDistance := NewDistance;
                        Supplier := CurrentBase;
                END IF;
        END IF;
        END IF;
        END IF;
END FOR;

RETURN(Supplier);
END METHOD;

{-------------------------------------------------------------------------}
ASK METHOD PickBestBase(IN Base1: BaseObj; IN Group : BaseQObj; INOUT
                                        Mode : INTEGER) : BaseObj;
{-------------------------------------------------------------------------}

{RETURNS CLOSEST COMMUNICATING BASE FOR ROUTE BUILDING, BASED ON MODE OF
TRANSPORTATION}

VAR

i, numItems : INTEGER;
ClosestBase : BaseObj;
CurrentBase : BaseObj;
BestDistance, NewDistance : REAL;

BEGIN
ClosestBase := NILOBJ;
numItems := ASK Group numberIn;

IF (Group = POEQ) OR (Group = ConusQ)
        IF (Mode < 4)                           {IE AIRCRAFT}

        FOR i := 1 TO numItems
                CurrentBase := ASK Group TO Remove;
                ASK Group TO Add(CurrentBase);
                IF CurrentBase.Name <> "Supply"
                        NewDistance :=  CalcDistance(Base1.Position,
                                                CurrentBase.Position);
                        IF (CurrentBase.HasAirPort) AND
                                        Communicates(CurrentBase,Base1);
                                IF ClosestBase = NILOBJ
                                        ClosestBase := CurrentBase;
                                        BestDistance := NewDistance
                                ELSIF NewDistance < BestDistance
                                        BestDistance := NewDistance;
                                        ClosestBase := CurrentBase;
                                END IF;
                        END IF;
                END IF;
        END FOR;

        ELSE
        FOR i := 1 TO numItems
                CurrentBase := ASK Group TO Remove;
                ASK Group TO Add(CurrentBase);
                IF CurrentBase.Name <> "Supply"
                        NewDistance :=  CalcDistance(Base1.Position,
                                                        CurrentBase.Position);
                        IF CurrentBase.HasSeaPort AND
                                                Communicates(CurrentBase,Base1)
                                IF ClosestBase = NILOBJ
                                        ClosestBase := CurrentBase;
                                        BestDistance := NewDistance
                                ELSIF NewDistance < BestDistance
                                        BestDistance := NewDistance;
                                        ClosestBase := CurrentBase;
                                END IF;
                        END IF;
                END IF;
        END FOR;

        END IF;

ELSE

        IF Mode < 4

        FOR i := 1 TO numItems
                CurrentBase := ASK Group TO Remove;
                ASK Group TO Add(CurrentBase);
                IF CurrentBase.Name <> "Supply"
                        NewDistance :=  CalcDistance(Base1.Position,
                                                       CurrentBase.Position);
                        IF (CurrentBase.HasAirPort) AND
                                                Communicates(CurrentBase,Base1);
                                IF ClosestBase = NILOBJ
                                        ClosestBase := CurrentBase;
                                        BestDistance := NewDistance
                                ELSIF NewDistance < BestDistance
                                        BestDistance := NewDistance;
                                        ClosestBase := CurrentBase;
                                END IF;
                        END IF;
                END IF;
        END FOR;

        END IF;

        IF (Mode > 3) OR (ClosestBase = NILOBJ)
        Mode := 4;

        FOR i := 1 TO numItems
                CurrentBase := ASK Group TO Remove;
                ASK Group TO Add(CurrentBase);
                IF CurrentBase.Name <> "Supply"

                        NewDistance :=  CalcDistance(Base1.Position,
                                                    CurrentBase.Position);
                        IF CurrentBase.HasSeaPort AND
                                             Communicates(CurrentBase,Base1)
                                IF ClosestBase = NILOBJ
                                        ClosestBase := CurrentBase;
                                        BestDistance := NewDistance
                                ELSIF NewDistance < BestDistance
                                        BestDistance := NewDistance;
                                        ClosestBase := CurrentBase;
                                END IF;
                        END IF;
                END IF;
        END FOR;

        END IF;
END IF;

IF ClosestBase = NILOBJ
        OUTPUT("     PROGRAM HALTED.  THERE IS A ROUTING INFEASIBILITY.");
        HALT;
END IF;

RETURN(ClosestBase);
END METHOD;


{---------------------------------------------------------------------}
{x} ASK METHOD PickClosestBaseInGroup(IN Base1: BaseObj; IN Group : BaseQObj) :
                                                                    BaseObj;
{---------------------------------------------------------------------}

{RETURNS THE CLOSEST BASE IN A GIVEN GROUP}

VAR

i, numItems : INTEGER;
ClosestBase : BaseObj;
CurrentBase : BaseObj;
BestDistance, NewDistance : REAL;

BEGIN
ClosestBase := NILOBJ;
numItems := ASK Group numberIn;

FOR i := 1 TO numItems
        CurrentBase := ASK Group TO Remove;
        ASK Group TO Add(CurrentBase);
        IF CurrentBase.Name <> "Supply"
        NewDistance :=  CalcDistance(Base1.Position, CurrentBase.Position);

        IF ClosestBase = NILOBJ
                ClosestBase := CurrentBase;
                BestDistance := NewDistance
        ELSIF NewDistance < BestDistance
                BestDistance := NewDistance;
                ClosestBase := CurrentBase;
        END IF;
        END IF;
END FOR;

RETURN(ClosestBase);
END METHOD;

{------------------------------------------------------------------------}
ASK METHOD BuildShipment(IN Requester : BaseObj; IN Item : CommodityObj; IN
                                              Route : BaseQObj) : ShipmentObj;
{------------------------------------------------------------------------}

{RETURNS CONSTRUCTED SHIPMENT}

VAR

Shipment : ShipmentObj;

BEGIN
{
OUTPUT("IN BUILDSHIPMENT");
}
NEW(Shipment);
ASK Shipment TO SetDestination(Requester);
ASK Shipment TO SetRoute(Route);
ASK Route TO Empty;
DISPOSE(Route);
ASK Shipment TO SetItem(Item);
RETURN(Shipment);



END METHOD;


{------------------------------------------------------------------------}
 ASK METHOD BuildRoute(IN Origin : BaseObj; IN Destination : BaseObj; IN Item
                            : CommodityObj; IN Surge : BOOLEAN) : BaseQObj;
{------------------------------------------------------------------------}

{RETURNS VALID ROUTE IF ONE IS AVAILABLE}

VAR

POEBase : BaseObj;
TheaterBase : BaseObj;
ILocBase : BaseObj;
Route : BaseQObj;
Mode : INTEGER;

BEGIN

Mode := ASK Item Priority;

NEW(Route);
IF Surge
        TheaterBase := PickBestBase(Destination, PODQ, Mode);
        POEBase := PickBestBase(Origin, POEQ, Mode);
        ASK Route TO Add(Origin);
        IF NOT(ASK Route Includes(POEBase))
```

```
                  ASK Route TO Add(POEBase);
          END IF;
          IF NOT(ASK Route Includes(TheaterBase))
                  ASK Route TO Add(TheaterBase);
          END IF;
          IF NOT(ASK Route Includes(Destination))
                  ASK Route TO Add(Destination);
          END IF;

  ELSIF Origin.Name = "Supply"
          ASK Route TO Add(Origin);
  ELSE
          CASE Origin.Group

          WHEN CONUS:
                  CASE Destination.Group

                  WHEN UNIT:
                          TheaterBase := PickBestBase(Destination,POSQ,Mode);
                          POEBase := PickBestBase(Origin,ConusQ,Mode);

                  WHEN THEATER:
                          TheaterBase := Destination;
                          POEBase := PickBestBase(Origin,ConusQ,Mode);
                  OTHERWISE {ILoc}
                          TheaterBase := Destination;
                          POEBase := PickBestBase(Origin,ConusQ,Mode);

                  END CASE;

                  ASK Route TO Add(Origin);
                  IF NOT(ASK Route Includes(POEBase))
                          ASK Route TO Add(POEBase);
                  END IF;
                  IF NOT(ASK Route Includes(TheaterBase))
                          ASK Route TO Add(TheaterBase);
                  END IF;
                  IF NOT(ASK Route Includes(Destination))
                          ASK Route TO Add(Destination);
                  END IF;

          WHEN ILOC:
                  CASE Destination.Group

                  WHEN UNIT:
                          TheaterBase :=
                                  PickBestBase(Destination,POSQ,Mode);

                  OTHERWISE {Theater}
                          TheaterBase := Destination;

                  END CASE;

                  ASK Route TO Add(Origin);
                  IF NOT(ASK Route Includes(TheaterBase))
                          ASK Route TO Add(TheaterBase);
                  END IF;
                  IF NOT(ASK Route Includes(Destination))
                          ASK Route TO Add(Destination);
                  END IF;
```

```
        OTHERWISE {Theater TO Unit}
                ASK Route TO Add(Origin);
                IF NOT(ASK Route Includes(Destination))
                        ASK Route TO Add(Destination);
                END IF;

        END CASE;

END IF;
        ASK Route TO RemoveThis(Route.First);
RETURN(Route);
END METHOD;

{
{---------------------------------------------------------------------------}
ASK METHOD PickBestILoc(IN ConusBase : BaseObj; IN TheaterBase : BaseObj) :
                                                                 BaseObj;
{---------------------------------------------------------------------------}

{RETURNS CLOSEST ILOC FOR "VIA" ROUTING (NORMAL SUPPLY)}

VAR

i, numItems : INTEGER;
CurrentILoc, BestILoc : BaseObj;
BestDistance, NewDistance, dist1, dist2 : REAL;
BEGIN

BestILoc := NILOBJ;
numItems := ASK ILocQ numberIn;
FOR i := 1 TO numItems
        CurrentILoc := ASK ILocQ TO Remove;
        ASK ILocQ TO Add(CurrentILoc);
        dist1 := CalcDistance(ConusBase.Position, CurrentILoc.Position);
        dist2 := CalcDistance(TheaterBase.Position, CurrentILoc.Position);
        NewDistance := dist1 + dist2;
        IF BestILoc = NILOBJ
                BestILoc := CurrentILoc;
                BestDistance := NewDistance;
        ELSIF NewDistance < BestDistance
                BestILoc := CurrentILoc;
                BestDistance := NewDistance;
        END IF;
END FOR;
RETURN(BestILoc)

END METHOD;
}

{---------------------------------------------------------------------------}
ASK METHOD ObjInit;
{---------------------------------------------------------------------------}
VAR

BEGIN

NEW(ConusQ);
NEW(ILocQ);
NEW(TheaterQ);
NEW(UnitQ);
NEW(POEQ);
NEW(PODQ);
NEW(POSQ);
NEW(DeadShipmentsQ);

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{-----------------------------------------------------------------------}
VAR

BEGIN
ASK ConusQ TO Empty;
DISPOSE(ConusQ);
ASK ILocQ TO Empty;
DISPOSE(ILocQ);
ASK TheaterQ TO Empty;
DISPOSE(TheaterQ);
ASK UnitQ TO Empty;
DISPOSE(UnitQ);
ASK POEQ TO Empty;
DISPOSE(POEQ);
ASK PODQ TO Empty;
DISPOSE(PODQ);
ASK POSQ TO Empty;
DISPOSE(POSQ);
DISPOSE(DeadShipmentsQ);

END METHOD;

END OBJECT;

{
{=======================================================================}
OBJECT ObjectNameObj;
{=======================================================================}

     {METHODS}

{-----------------------------------------------------------------------}
ASK METHOD
{-----------------------------------------------------------------------}
VAR

BEGIN

END METHOD;

END OBJECT;
}

{-----------------------------------------------------------------------}
PROCEDURE Communicates(IN Base1, Base2 : BaseObj) : BOOLEAN;
{-----------------------------------------------------------------------}
VAR

boolean : BOOLEAN;
```

```
BEGIN

IF (Base1.HasSeaPort) AND (Base2.HasSeaPort)
        boolean := TRUE;

ELSIF (Base1.HasAirPort) AND (Base2.HasAirPort)
        boolean := TRUE;

ELSIF (Base1.HasRail) AND (Base2.HasRail) AND (ASK Base1.RailYard.Network TO
 Includes(Base2))
        boolean := TRUE;

ELSIF (Base1.HasTruckStop) AND (Base2.HasTruckStop) AND (ASK
 Base1.TruckStop.Network TO Includes(Base2))
        boolean := TRUE;

ELSE
        boolean := FALSE;

END IF;

RETURN(boolean);

END PROCEDURE;

END MODULE.
```

```
DEFINITION MODULE MyQueue;

{contains the root objects NamedObj, an OBJECT with a Name field, and
MyQueueObj, a QueueObj that can find NamedObjs by their Name}

{Import statements}

FROM GrpMod IMPORT QueueObj;

{Type Declarations}

TYPE

{=========================================================================}
NamedObj = OBJECT {=========================================================
        {fields}
        Name : STRING;
        {ASK METHOD ObjInit;}
        ASK METHOD InputName;
        ASK METHOD SetName(IN NewName : STRING);
        ASK METHOD ObjTerminate;
END OBJECT;

{=========================================================================}
MyQueueObj = PROTO(QueueObj[ANYOBJ : #NamedObj])
{=========================================================================}
     {fields}
     {CurrentObj : ANYOBJ;}

     {methods}
     ASK METHOD FindByName(IN Name : STRING) : #NamedObj;
     ASK METHOD Display(INOUT j : INTEGER);
{
     ASK METHOD Empty;
}
OVERRIDE
     ASK METHOD ObjTerminate;

     ASK METHOD Empty;

END PROTO;

END MODULE.
```

```
IMPLEMENTATION MODULE MyQueue;

{MyQObject}

{Import statements}

FROM SimMod IMPORT InterruptAll,
                   NumActivities;
FROM GrpMod IMPORT QueueObj;
FROM SOUTPUT IMPORT SOUTPUT;
FROM Trash IMPORT GarbageDisposal,
                  TrashCan;
{
FROM IMPORT ;
}
{Definitions}

{=============================================================================}
OBJECT NamedObj;
{=============================================================================}
        {METHODS}
{-----------------------------------------------------------------------------}
        {ASK METHOD ObjInit;
{-----------------------------------------------------------------------------}

        BEGIN
        END METHOD;}

{-----------------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{-----------------------------------------------------------------------------}

BEGIN

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD InputName;
{-----------------------------------------------------------------------------}
VAR

string : STRING;

BEGIN

OUTPUT("     Input Name.");
INPUT(string);
ASK SELF TO SetName(string);

END METHOD;

{-----------------------------------------------------------------------------}
        ASK METHOD SetName(IN NewName : STRING);
{-----------------------------------------------------------------------------}

BEGIN

Name := NewName;

END METHOD;
END OBJECT;

{=========================================================================}
PROTO MyQueueObj;
{=========================================================================}

     {METHODS}
{-------------------------------------------------------------------------}
     ASK METHOD FindByName(IN FindName : STRING) : NamedObj;
{-------------------------------------------------------------------------}

{RETURNS NAMED OBJECT.  RETURNS NILOBJ IF NO OBJECT WITH DESIRED NAME.}

VAR
CurrentObj : NamedObj;
ReturnObj : NamedObj;
i, numItems : INTEGER;

BEGIN

ReturnObj := NILOBJ;
numItems := numberIn;
FOR i := 1 TO numItems
        CurrentObj := Remove;
        Add(CurrentObj);
        IF CurrentObj.Name = FindName
                ReturnObj := CurrentObj;
        END IF;
END FOR;
RETURN(ReturnObj);

END METHOD;

{-------------------------------------------------------------------------}
ASK METHOD Display(INOUT j : INTEGER);
{-------------------------------------------------------------------------}

{DISPLAYS NAMES OF THE MEMBERS OF THE QUEUE IN THREE COLUMN FORMAT}


CONST

format = "     ***************     ***************     *************** ";

VAR

string, string1, string2, string3 : STRING;
object, lastobject : NamedObj;
name : STRING;

BEGIN

SOUTPUT(" ",j);

IF ASK SELF numberIn > 0
        lastobject := ASK  SELF Last;
        LOOP
                string1 := "              ";
                string2 := "               ";
                string3 := "               ";

                object := ASK SELF TO Remove;
                ASK SELF TO Add(object);
                name := ASK object Name;
                string1 := name;
                IF object = lastobject
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;

                object := ASK SELF TO Remove;
                ASK SELF TO Add(object);
                name := ASK object Name;
                string2 := name;
                IF object = lastobject
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;

                object := ASK SELF TO Remove;
                ASK SELF TO Add(object );
                name  := ASK object Name;
                string3 := name;
                IF object  = lastobject
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;
                string := SPRINT(string1,string2,string3) WITH format;
                SOUTPUT(string,j);

        END LOOP;

END IF;
SOUTPUT(" ",j);

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD Empty;
{--------------------------------------------------------------------------}

{EMPTYS QUEUE}

VAR

i, numItems : INTEGER;
object : NamedObj;

BEGIN

numItems := numberIn;
FOR i := 1 TO numItems
        object := Remove;
END FOR
END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{-----------------------------------------------------------------------------}
VAR

i, j, numItems : INTEGER;
object : ANYOBJ;

BEGIN

numItems := numberIn;
FOR i := 1 TO numItems

        object := ASK SELF TO Remove;
        j := NumActivities(object);
        IF j > 0
                InterruptAll(object);
                ASK TrashCan TO Add(object);
        ELSE
                IF NOT (ASK GarbageDisposal Includes(object))
                        ASK GarbageDisposal TO Add(object);
                END IF;
        END IF;

END FOR;
INHERITED ObjTerminate;

END METHOD;

END PROTO;

END MODULE.
```

```
IMPLEMENTATION MODULE Node;

{Node Object}

{Import statements}

FROM CommodQ IMPORT CommodityQObj,
                    CommodityObj;
FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM Distant IMPORT PositionRecType;

{Type Declarations}

{===========================================================================}
OBJECT NodeObj;
{===========================================================================}

{METHODS}

{---------------------------------------------------------------------------}
 ASK METHOD ObjInit;
{---------------------------------------------------------------------------}
{should read from file describing node and enter node fields except for links be
BEGIN
        NEW(Inventory);
        NEW(Position);
END METHOD;

{---------------------------------------------------------------------------}
 ASK METHOD ObjTerminate;
{---------------------------------------------------------------------------}
BEGIN
        INHERITED ObjTerminate;
        DISPOSE(Position);
        DISPOSE(Inventory);
END METHOD;

{---------------------------------------------------------------------------}
 ASK METHOD SetPosition(IN NewPosition : PositionRecType);
{---------------------------------------------------------------------------}
BEGIN
        Position := CLONE(NewPosition);

END METHOD;

END OBJECT;

END MODULE.
```

```
DEFINITION MODULE Node;

{Definition of simple node object, a refinement of the NamedObj.  The NodeObj ha

{Import statements}

FROM CommodQ IMPORT CommodityQObj,
                    CommodityObj;
FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM Distant IMPORT PositionRecType;

{Type Declarations}

TYPE

{=========================================================================}
NodeObj = OBJECT(NamedObj)
{=========================================================================}

{Fields}

     Position : PositionRecType;

     Inventory :  CommodityQObj;


{Methods}

        ASK METHOD SetPosition(IN NewPosition : PositionRecType);


        ASK METHOD ObjInit;
OVERRIDE
        ASK METHOD ObjTerminate;
END OBJECT;

{=========================================================================}
NodeQObj = PROTO(MyQueueObj[ANYOBJ : NodeObj]);
{=========================================================================}
END PROTO;

END MODULE.
```

```
DEFINITION MODULE Port;

{Port Object is a auxilary for the BaseObj.  PortObjs do all the Cargo handling
and Transporter loading functions for the BaseObj they belong to}

{Import statements}

FROM Trnsprt IMPORT ALL TransporterClassType,
                    ALL TransporterSubClassType,
                        TransporterObj,
                        TransporterQObj,
                        LoadQObj;
FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj;
FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM GrpMod IMPORT QueueObj;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;

{Type Declarations}

TYPE

{============================================================================}
CargoGroupObj = OBJECT(LoadQObj)
{============================================================================}

Destination : BaseObj;
Priority : INTEGER;
Owner : PortObj;
OverSize : BOOLEAN;
PaxTally : REAL;
GasTally : REAL;
CubeTally : REAL;
AreaTally : REAL;

ASK METHOD SetDestination(INOUT NewDestination : BaseObj);
ASK METHOD SetPriority;
ASK METHOD SetOwner(INOUT NewOwner : PortObj);
ASK METHOD ResetTallys;
ASK METHOD SetOverSize(IN NewOverSize : BOOLEAN);
ASK METHOD CheckSize;
ASK METHOD OrderTransporter;
ASK METHOD AddFirstTime(IN NewMember : ShipmentObj);

OVERRIDE
ASK METHOD Add(IN NewMember : ShipmentObj);
ASK METHOD Remove : ShipmentObj;
ASK METHOD RemoveThis(IN member : ShipmentObj);

END OBJECT;

{============================================================================}
LoadingDockObj = OBJECT(QueueObj[ANYOBJ : CargoGroupObj])
{============================================================================}
ASK METHOD FindHighPri() : CargoGroupObj;
OVERRIDE
ASK METHOD ObjTerminate;
```

```
END OBJECT;

{========================================================================}
PortObj = OBJECT
{========================================================================}

{FIELDS}

Owner : BaseObj;
Class : TransporterClassType;

MaxCapacity : INTEGER;
MaxSize : REAL;

Network : BaseQObj;

ArrivalsQ : TransporterQObj;
BerthsQ : TransporterQObj;
ParkedQ : TransporterQObj;

LoadingDock : LoadingDockObj;
OverSizeLoadingDock : LoadingDockObj;
PaxLoadingDock : LoadingDockObj;
GasLoadingDock : LoadingDockObj;

{METHODS}

ASK METHOD SetOwner(INOUT NewOwner : BaseObj);
ASK METHOD SetClass(IN NewClass : STRING);
ASK METHOD SetMaxCapacity(IN NewMaxCapacity : INTEGER);
ASK METHOD SetMaxSize(IN NewMaxSize : REAL);

ASK METHOD GetArrival(INOUT NewArrival : TransporterObj);
ASK METHOD GetDeparture(INOUT NewDeparture : TransporterObj);
ASK METHOD CheckBerths;

ASK METHOD SortCargo(INOUT Shipment : ShipmentObj);
ASK METHOD RequestTransporter(IN SubClass : TransporterSubClassType; IN OverSize
         : BOOLEAN);
ASK METHOD Load(INOUT Transporter : TransporterObj);
ASK METHOD LoadPax(INOUT Destination : BaseObj; INOUT Transporter :
         TransporterObj; INOUT Load : LoadQObj);
ASK METHOD LoadGas(INOUT Destination : BaseObj; INOUT Transporter :
         TransporterObj; INOUT Load : LoadQObj);
ASK METHOD LoadCargo(INOUT Destination : BaseObj; INOUT Transporter :
         TransporterObj; INOUT Load : LoadQObj);
ASK METHOD LoadItem(INOUT Shipment : ShipmentObj; IN Transporter :
         TransporterObj; INOUT Load : LoadQObj; INOUT CurrentGroup :
         CargoGroupObj);

ASK METHOD ObjInit;
ASK METHOD ObjTerminate;

END OBJECT;

END MODULE.
```

```
IMPLEMENTATION MODULE Port;

{Generic Port Object}

{Import statements}

FROM Trnsprt IMPORT ALL TransporterClassType,
                    ALL TransporterSubClassType,
                        TransporterObj,
                        TransporterQObj,
                        LoadQObj,
                        PalletHeight;
FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM CommodQ IMPORT ALL CommodityClassType,
                    CommodityObj,
                    CommodityQObj;
FROM TManage IMPORT TransporterManager;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;

{Definitions}


{=====================================================================}
OBJECT CargoGroupObj;
{=====================================================================}

{Methods}

{---------------------------------------------------------------------}
ASK METHOD SetDestination(INOUT NewDestination : BaseObj);
{---------------------------------------------------------------------}
VAR

BEGIN

Destination := NewDestination;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD SetOwner(INOUT NewOwner : PortObj);
{---------------------------------------------------------------------}
VAR

BEGIN

Owner := NewOwner;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD SetOverSize(IN NewOverSize : BOOLEAN);
{---------------------------------------------------------------------}
VAR

BEGIN
```

```
OverSize := NewOverSize;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetPriority;
{-----------------------------------------------------------------------}
VAR

CurrentShipment : ShipmentObj;
HighPri : INTEGER;
i, j, numItems : INTEGER;

BEGIN
HighPri := 13;
numItems := numberIn;
FOR i := 1 TO numItems;

        CurrentShipment := ASK SELF Remove;
        ASK SELF TO Add(CurrentShipment);
        IF HighPri > ASK CurrentShipment.Item Priority
                HighPri := ASK CurrentShipment.Item Priority;
        END IF;
END FOR;
Priority := HighPri;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD ResetTallys;
{-----------------------------------------------------------------------}

VAR

BEGIN

IF TotalPax < 1.0
        PaxTally := 0.0;
END IF;
IF TotalGas < 1.0
        GasTally := 0.0;
END IF;
IF TotalCube < 1.0
        CubeTally := 0.0;
END IF;
IF TotalArea < 1.0
        AreaTally := 0.0;
END IF;
IF CubeTally < 0.0
        CubeTally := 0.0;
END IF;
IF AreaTally < 0.0
        AreaTally := 0.0;
END IF;
IF GasTally < 0.0
        GasTally := 0.0;
END IF;
IF PaxTally < 0.0
        PaxTally := 0.0;
END IF;
END METHOD;

{------------------------------------------------------------------------}
ASK METHOD CheckSize;
{------------------------------------------------------------------------}

{REQUESTS TRANSPORTER IF LOADING DOCK CAPACITY EXCEED NOMINAL TRANSPORTATION
ALREADY ON ORDER}

VAR

BEGIN

CASE Owner.Class
WHEN Aircraft:
        IF TotalPax - PaxTally > 75.0
                ASK Owner TO RequestTransporter(Pax, OverSize);
                PaxTally := PaxTally + 150.0;
        END IF;
        IF TotalGas - GasTally > 1000.0
                ASK Owner TO RequestTransporter(Liquid, OverSize);
                GasTally := GasTally + 1000.0;
        END IF;
WHEN Rail:
        IF TotalArea - AreaTally > 14000.00
                ASK Owner TO RequestTransporter(RoRo, OverSize);
                AreaTally := AreaTally + 28350.0;
                CubeTally := CubeTally + 274000.00;
        END IF;
        IF TotalPax - PaxTally > 10.0
                ASK Owner TO RequestTransporter(Pax, OverSize);
                PaxTally := PaxTally + 770.0;
        END IF;
        IF TotalGas - GasTally > 476.00
                ASK Owner TO RequestTransporter(Liquid, OverSize);
                GasTally := GasTally + 14000.0;
        END IF;
WHEN Truck:
        IF  (TotalGas - GasTally > 500.00)
                ASK Owner TO RequestTransporter(Liquid, OverSize);
                GasTally := GasTally + 1190.00;
        END IF;
WHEN Ship:
        IF TotalGas - GasTally > 1000.00
                ASK Owner TO RequestTransporter(Liquid, OverSize);
                GasTally := GasTally + 100000.0;
        END IF;
        IF TotalArea - AreaTally > 30000.00
                        ASK Owner TO RequestTransporter(RoRo, OverSize);
                CubeTally := CubeTally + 772000.0;
                AreaTally := AreaTally + 115157.00;
                GasTally := GasTally + 1000.00;
                PaxTally := PaxTally + 10.00;
        END IF;
OTHERWISE
END CASE;

END METHOD;
{------------------------------------------------------------------------}
```

```
ASK METHOD OrderTransporter;
{--------------------------------------------------------------------}

{REQUESTS TRANSPORTER IF LOADING DOCK CAPACITY EXCEED NOMINAL TRANSPORTATION
ALREADY ON ORDER}

VAR

BEGIN

OUTPUT("     IN OrderTransporter - ", Owner.Class);
OUTPUT("     TotalCube = ", TotalCube, " CubeTally = ", CubeTally);
OUTPUT("     TotalArea = ", TotalArea, " AreaTally = ", AreaTally);
OUTPUT("     TotalPax = ", TotalPax, " PaxTally = ", PaxTally);
OUTPUT("     TotalGas = ", TotalGas, " GasTally = ", GasTally);

CASE Owner.Class
WHEN Aircraft:
        IF (TotalCube - CubeTally >= 1.00)
                ASK Owner TO RequestTransporter(General, OverSize);
                CubeTally := CubeTally + 3350.0;
                PaxTally := PaxTally + 75.0;
                GasTally := GasTally + 100.0;
        ELSIF (TotalPax - PaxTally >= 1.0) AND (TotalPax - PaxTally <= 75.0)
                ASK Owner TO RequestTransporter(General, OverSize);
                CubeTally := CubeTally + 3350.0;
                PaxTally := PaxTally + 75.0;
        ELSIF (TotalGas - GasTally >= 1.00) AND (TotalGas - GasTally <= 100.0)
                ASK Owner TO RequestTransporter(General, OverSize);
                CubeTally := CubeTally + 3350.0;
                GasTally := GasTally + 100.0;
                PaxTally := PaxTally + 0.00;
        END IF;
WHEN Rail:
        IF (TotalArea - AreaTally >= 1.00) AND (TotalArea - AreaTally <=
         16000.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 150000.00;
                AreaTally := AreaTally + 16879.00;
                GasTally := GasTally + 476.00;
                PaxTally := PaxTally + 10.00;
        ELSIF (TotalGas - GasTally >= 1.00) AND (TotalGas - GasTally <= 476.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                AreaTally := AreaTally + 16879.00;
                CubeTally := CubeTally + 150000.00;
                GasTally := GasTally + 476.00;
                PaxTally := PaxTally + 10.00;
        ELSIF (TotalPax - PaxTally > 1.00) AND (TotalPax - PaxTally <= 10.0)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 150000.00;
                AreaTally := AreaTally + 16879.00;
                GasTally := GasTally + 476.00;
                PaxTally := PaxTally + 10.00;
        END IF;
WHEN Truck:
        IF (TotalCube - CubeTally >= 1.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 13000.00;
                GasTally := GasTally + 500.00;
                PaxTally := PaxTally + 200.00;
        ELSIF (TotalGas - GasTally >= 1.00) AND (TotalGas - GasTally <= 500.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 13000.00;
                GasTally := GasTally + 500.00;
                PaxTally := PaxTally + 200.00;
        ELSIF (TotalPax - PaxTally >= 1.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 13000.00;
                GasTally := GasTally + 500.00;
                PaxTally := PaxTally + 200.00;
        END IF;
WHEN Ship:
        IF (TotalArea - AreaTally >= 1.00) AND (TotalArea - AreaTally <=
         36000.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 602120.0;
                AreaTally := AreaTally + 36000.0;
                GasTally := GasTally + 1000.00;
                PaxTally := PaxTally + 10.00;
        ELSIF (TotalGas - GasTally >= 1.00) AND (TotalGas - GasTally <=
         1000.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 602120.0;
                AreaTally := AreaTally + 36000.0;
                GasTally := GasTally + 1000.00;
                PaxTally := PaxTally + 10.00;
        ELSIF (TotalCube - CubeTally >= 1.00) AND (TotalCube - CubeTally <=
         602120.00)
                ASK Owner TO RequestTransporter(BreakBulk, OverSize);
                CubeTally := CubeTally + 602120.0;
                AreaTally := AreaTally + 36000.0;
                GasTally := GasTally + 1000.00;
                PaxTally := PaxTally + 10.00;
        END IF;

END CASE;
CheckSize;
END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD AddFirstTime(IN NewMember : ShipmentObj);
{----------------------------------------------------------------------------}
VAR

BEGIN

INHERITED Add(NewMember);

ASK SELF TO OrderTransporter;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD Add(IN NewMember : ShipmentObj);
{----------------------------------------------------------------------------}
VAR

BEGIN

INHERITED Add(NewMember);
```

```
CASE NewMember.Item.Class
WHEN Personnel:
        PaxTally := PaxTally + NewMember.Item.OnHand;
WHEN Fuel:
        GasTally := GasTally + NewMember.Item.OnHand;
WHEN Major:
        AreaTally := AreaTally + (NewMember.Item.OnHand * NewMember.Item.Length
                                  * NewMember.Item.Width )/ 144.00;
        CubeTally := CubeTally + (NewMember.Item.OnHand * NewMember.Item.Length
                 * NewMember.Item.Width * PalletHeight)/1728.00;


OTHERWISE
        CubeTally := CubeTally + (NewMember.Item.OnHand * NewMember.Item.Length
                 * NewMember.Item.Width * NewMember.Item.Height)/1728.00;
        AreaTally := AreaTally + ((NewMember.Item.OnHand * NewMember.Item.Length
                 * NewMember.Item.Width * NewMember.Item.Height) / PalletHeight)
                 / 144.00;
END CASE;


END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD Remove : ShipmentObj;
{-----------------------------------------------------------------------}

VAR
Shipment : ShipmentObj;

BEGIN

Shipment := INHERITED Remove;

CASE Shipment.Item.Class
WHEN Personnel:
        PaxTally := PaxTally - Shipment.Item.OnHand;
WHEN Fuel:
        GasTally := GasTally - Shipment.Item.OnHand;
WHEN Major:
        AreaTally := AreaTally - (Shipment.Item.OnHand * Shipment.Item.Length
                                  * Shipment.Item.Width) / 144.00;
        CubeTally := CubeTally - (Shipment.Item.OnHand * Shipment.Item.Length
                 * Shipment.Item.Width * PalletHeight)/1728.00;
OTHERWISE
        CubeTally := CubeTally - (Shipment.Item.OnHand * Shipment.Item.Length *
                         Shipment.Item.Width * Shipment.Item.Height)/1728.00;
        AreaTally := AreaTally - ((Shipment.Item.OnHand * Shipment.Item.Length
                 * Shipment.Item.Width * Shipment.Item.Height) / PalletHeight)
                 / 144.00;
END CASE;
{
IF CubeTally < 0.0
        CubeTally := 0.0;
END IF;
IF AreaTally < 0.0
        AreaTally := 0.0;
END IF;
IF GasTally < 0.0
        GasTally := 0.0;
END IF;
IF PaxTally < 0.0
        PaxTally := 0.0;
END IF;
}

RETURN(Shipment);

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD RemoveThis(IN member : ShipmentObj);
{-----------------------------------------------------------------------------}

VAR
BEGIN

IF ASK SELF Includes(member)

        CASE member.Item.Class
        WHEN Personnel:
                PaxTally := PaxTally - member.Item.OnHand;
        WHEN Fuel:
                GasTally := GasTally - member.Item.OnHand;
        WHEN Major:
                AreaTally := AreaTally - (member.Item.OnHand *
                         member.Item.Length * member.Item.Width) / 144.00;
                CubeTally := CubeTally - (member.Item.OnHand *
                         member.Item.Length * member.Item.Width *
                         PalletHeight)/1728.00;
        OTHERWISE
                CubeTally := CubeTally - (member.Item.OnHand *
                         member.Item.Length * member.Item.Width *
                         member.Item.Height) / 1728.00;
                AreaTally := AreaTally - ((member.Item.OnHand *
                         member.Item.Length * member.Item.Width *
                         member.Item.Height) / PalletHeight) / 144.00;
        END CASE;
END IF;
{
IF CubeTally < 0.0
        CubeTally := 0.0;
END IF;
IF AreaTally < 0.0
        AreaTally := 0.0;
END IF;
IF GasTally < 0.0
        GasTally := 0.0;
END IF;
IF PaxTally < 0.0
        PaxTally := 0.0;
END IF;
}

INHERITED RemoveThis(member);

END METHOD;
END OBJECT;

{=====================================================================}
OBJECT LoadingDockObj;
{=====================================================================}

{---------------------------------------------------------------------}
ASK METHOD FindHighPri() : CargoGroupObj;
{---------------------------------------------------------------------}

{FINDS HIGHEST PRIORITY CARGOGROUP ON LOADING DOCK}

VAR

CurrentGroup : CargoGroupObj;
HighPri : INTEGER;
BestGroup : CargoGroupObj;
i, j, numItems : INTEGER;

BEGIN

{
OUTPUT("IN FindHighPri - ");
}
numItems := numberIn;
BestGroup := NILOBJ;
HighPri := 13;
FOR i := 1 TO numItems;
        CurrentGroup := ASK SELF Remove;
        ASK SELF TO Add(CurrentGroup);
        ASK CurrentGroup TO SetPriority;
        IF HighPri > ASK CurrentGroup Priority
                HighPri := ASK CurrentGroup Priority;
                BestGroup := CurrentGroup;
        END IF;
END FOR;
RETURN(BestGroup);

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{---------------------------------------------------------------------}
VAR

i, j, numItems : INTEGER;
object : CargoGroupObj;

BEGIN
numItems := numberIn;
FOR i := 1 TO numItems

        object := ASK SELF TO Remove;
        DISPOSE(object);
END FOR;
INHERITED ObjTerminate;

END METHOD;

END OBJECT;
```

```
{=================================================================}
OBJECT PortObj;
{=================================================================}

     {METHODS}

{-----------------------------------------------------------------}
ASK METHOD SetClass(IN NewClass : STRING);
{-----------------------------------------------------------------}
VAR

BEGIN


CASE NewClass

        WHEN "Aircraft":
                Class := Aircraft;
        WHEN "Ship":
                Class := Ship;
        WHEN "Rail":
                Class := Rail;
        WHEN "Truck":
                Class := Truck;
        OTHERWISE
                OUTPUT("Port Class assignment out of range");
                HALT;
END CASE;


END METHOD;


{-----------------------------------------------------------------}
ASK METHOD SetOwner(INOUT NewOwner : BaseObj);
{-----------------------------------------------------------------}
VAR

BEGIN

Owner := NewOwner;

END METHOD;


{-----------------------------------------------------------------}
ASK METHOD SetMaxCapacity(IN NewMaxCapacity : INTEGER);
{-----------------------------------------------------------------}
VAR

BEGIN

MaxCapacity := NewMaxCapacity;

END METHOD;

{-----------------------------------------------------------------}
ASK METHOD SetMaxSize(IN NewMaxSize : REAL);
{-----------------------------------------------------------------}
```

```
VAR

BEGIN

MaxSize := NewMaxSize;

END METHOD;

{-----------------------------------------------------------------------}
     ASK METHOD GetArrival(INOUT NewArrival : TransporterObj);
{-----------------------------------------------------------------------}

{PLACES ARRIVING TRANSPRTER IN ARRIVALQ ANF CHECKS IN BERTH IS AVAILABLE}


VAR

BEGIN
{
OUTPUT("IN GetArrival - ", Class, " - ", Owner.Name);
}

ASK ArrivalsQ TO Add(NewArrival);
ASK SELF TO CheckBerths;

END METHOD;

{-----------------------------------------------------------------------}
     ASK METHOD CheckBerths;
{-----------------------------------------------------------------------}

{CHECKS IF BERTH IS AVAILABLE}

 VAR

CurrentArrival : TransporterObj;
i, numArrivals : INTEGER;

BEGIN

{
OUTPUT("IN CheckBerths - ", Class, " - ", Owner.Name);
}

numArrivals := ASK ArrivalsQ numberIn;
FOR i := 1 TO numArrivals
        CurrentArrival := ASK ArrivalsQ TO Remove;

        IF (MaxCapacity > ASK BerthsQ numberIn);
                ASK BerthsQ TO Add(CurrentArrival);
                TELL CurrentArrival TO Unload;
        ELSE
                ASK ArrivalsQ TO Add(CurrentArrival);
        END IF;
END FOR;

END METHOD;

{-----------------------------------------------------------------------}
```

```
     ASK METHOD GetDeparture(INOUT NewDeparture : TransporterObj);
{---------------------------------------------------------------------------}
{REMOVES DEPARTING TRANSPORTERS FROM QUEUES}

VAR

BEGIN
{
OUTPUT("IN GetDeparture - ", Class, " - ", Owner.Name);
}

IF ASK BerthsQ Includes(NewDeparture)
        ASK BerthsQ TO RemoveThis(NewDeparture);
END IF;
IF ASK ParkedQ Includes(NewDeparture)
        ASK ParkedQ TO RemoveThis(NewDeparture);
END IF;
ASK SELF TO CheckBerths;

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD SortCargo(INOUT Shipment : ShipmentObj); {---------------------

{SORTS SHIPMENTS INTO PROPER CARGO GROUPS}


VAR

Group, BestGroup : CargoGroupObj;
i, numItems : INTEGER;
Receiver : BaseObj;

BEGIN

OUTPUT("IN SortCargo - ", Class, " - ", Owner.Name);


Receiver := ASK Shipment.Route First;
BestGroup := NILOBJ;
CASE Shipment.Item.Class
WHEN Personnel:
        numItems := ASK PaxLoadingDock numberIn;
        FOR i := 1 TO numItems
                Group := ASK PaxLoadingDock Remove;
                ASK PaxLoadingDock TO Add(Group);
                IF (Group.Destination.Name = Receiver.Name)
                        BestGroup := Group;
                END IF;
        END FOR;
        IF BestGroup <> NILOBJ
                ASK BestGroup TO AddFirstTime(Shipment);
        ELSE
                NEW(BestGroup);
                ASK BestGroup TO SetDestination(Receiver);
                ASK BestGroup TO SetOwner(SELF);
                ASK BestGroup TO SetOverSize(FALSE);
                ASK BestGroup TO AddFirstTime(Shipment);
                ASK BestGroup TO SetPriority;
                ASK PaxLoadingDock TO Add(BestGroup);

        END IF;

WHEN Fuel:
        numItems := ASK GasLoadingDock numberIn;
        FOR i := 1 TO numItems
                Group := ASK GasLoadingDock Remove;
                ASK GasLoadingDock TO Add(Group);
                IF (Group.Destination.Name = Receiver.Name)
                        BestGroup := Group;
                END IF;
        END FOR;
        IF BestGroup <> NILOBJ
                ASK BestGroup TO AddFirstTime(Shipment);
        ELSE
                NEW(BestGroup);
                ASK BestGroup TO SetDestination(Receiver);
                ASK BestGroup TO SetOwner(SELF);
                ASK BestGroup TO SetOverSize(FALSE);
                ASK BestGroup TO AddFirstTime(Shipment);
                ASK BestGroup TO SetPriority;
                ASK GasLoadingDock TO Add(BestGroup);

        END IF;

OTHERWISE

IF (ASK Shipment.Item OverSize)
        numItems := ASK OverSizeLoadingDock numberIn;
        FOR i := 1 TO numItems
                Group := ASK OverSizeLoadingDock Remove;
                ASK OverSizeLoadingDock TO Add(Group);
                IF (Group.Destination.Name = Receiver.Name)
                        BestGroup := Group;
                END IF;
        END FOR;
        IF BestGroup <> NILOBJ
                ASK BestGroup TO AddFirstTime(Shipment);
        ELSE
                NEW(BestGroup);
                ASK BestGroup TO SetDestination(Receiver);
                ASK BestGroup TO SetOwner(SELF);
                ASK BestGroup TO SetOverSize(TRUE);
                ASK BestGroup TO AddFirstTime(Shipment);
                ASK BestGroup TO SetPriority;
                ASK OverSizeLoadingDock TO Add(BestGroup);

        END IF;

ELSE
        numItems := ASK LoadingDock numberIn;
        FOR i := 1 TO numItems
                Group := ASK LoadingDock Remove;
                ASK LoadingDock TO Add(Group);
                IF (Group.Destination.Name = Receiver.Name);
                        BestGroup := Group;
                END IF;
        END FOR;
        IF BestGroup <> NILOBJ
```

```
                ASK BestGroup TO AddFirstTime(Shipment);
                ASK BestGroup TO SetPriority;
        ELSE
                NEW(BestGroup);
                ASK BestGroup TO SetDestination(Receiver);
                ASK BestGroup TO SetOwner(SELF);
                ASK BestGroup TO SetOverSize(FALSE);
                ASK BestGroup TO SetPriority;
                ASK BestGroup TO AddFirstTime(Shipment);
                ASK LoadingDock TO Add(BestGroup);
        END IF;
END IF;
END CASE;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD RequestTransporter(IN SubClass : TransporterSubClassType; IN OverSize
         : BOOLEAN);
{----------------------------------------------------------------------------}

{PASSES REQUEST FROM CARGOGROUP TO TRANSPORTER MANAGER}

VAR

BEGIN

ASK TransporterManager TO ReceiveRequest(Owner, Class, SubClass, OverSize);

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD Load(INOUT Transporter : TransporterObj);
{----------------------------------------------------------------------------}

{BUILDS A PROPER SIZED LOAD FOR A GIVEN TRANSPORTER}

VAR

Group, OverSizeGroup, CurrentGroup : CargoGroupObj;
HighPri, OverSizeHighPri : INTEGER;
Load : LoadQObj;
Destination : BaseObj;

BEGIN


OUTPUT("IN Load - ", Class, " - ", Owner.Name);


NEW(Load);
                        {if transporter is oversize find hi pris in regular and
                        oversize cargo. if oversize has highest pri, start with
                        oversize if not start with normal.  Alternately check
                        oversize and normal queues until aircraft is loaded.}

CASE Transporter.SubClass
WHEN Pax:
        CurrentGroup := ASK PaxLoadingDock TO FindHighPri();
        IF CurrentGroup = NILOBJ
```

```
                IF Transporter.OverSize
                    OverSizeGroup :=  ASK OverSizeLoadingDock TO FindHighPri();
                    IF OverSizeGroup = NILOBJ
                            OverSizeHighPri := 13;
                    ELSE
                            OverSizeHighPri := ASK OverSizeGroup Priority;
                    END IF;

                    Group := ASK LoadingDock TO FindHighPri();
                    IF Group = NILOBJ
                            HighPri := 13;
                    ELSE
                            HighPri := ASK Group Priority;
                    END IF;
                                                    {Pick Highest Priority Group}


                    IF OverSizeHighPri <= HighPri
                            CurrentGroup := OverSizeGroup;
                    ELSE
                            CurrentGroup := Group;
                    END IF;

                ELSE
                    CurrentGroup := ASK LoadingDock TO FindHighPri();
                END IF;
            END IF;
            IF CurrentGroup = NILOBJ
                    CurrentGroup := ASK GasLoadingDock TO FindHighPri();
            END IF;

        WHEN Liquid:
            CurrentGroup := ASK GasLoadingDock TO FindHighPri();
            IF CurrentGroup = NILOBJ
                IF Transporter.OverSize
                    OverSizeGroup :=  ASK OverSizeLoadingDock TO FindHighPri();
                    IF OverSizeGroup = NILOBJ
                            OverSizeHighPri := 13;
                    ELSE
                            OverSizeHighPri := ASK OverSizeGroup Priority;
                    END IF;

                    Group := ASK LoadingDock TO FindHighPri();
                    IF Group = NILOBJ
                            HighPri := 13;
                    ELSE
                            HighPri := ASK Group Priority;
                    END IF;
                                                    {Pick Highest Priority Group}


                    IF OverSizeHighPri <= HighPri
                            CurrentGroup := OverSizeGroup;
                    ELSE
                            CurrentGroup := Group;
                    END IF;

                ELSE
                    CurrentGroup := ASK LoadingDock TO FindHighPri();
                END IF;
        END IF;
        IF CurrentGroup = NILOBJ
                CurrentGroup := ASK PaxLoadingDock TO FindHighPri();
        END IF;

OTHERWISE
        IF Transporter.OverSize

                OverSizeGroup :=  ASK OverSizeLoadingDock TO FindHighPri();
                IF OverSizeGroup = NILOBJ
                        OverSizeHighPri := 13;
                ELSE
                        OverSizeHighPri := ASK OverSizeGroup Priority;
                END IF;

                Group := ASK LoadingDock TO FindHighPri();
                IF Group = NILOBJ
                        HighPri := 13;
                ELSE
                        HighPri := ASK Group Priority;
                END IF;
                                                {Pick Highest Priority Group}


                IF OverSizeHighPri <= HighPri
                        CurrentGroup := OverSizeGroup;
                ELSE
                        CurrentGroup := Group;
                END IF;
        ELSE
                CurrentGroup := ASK LoadingDock TO FindHighPri();
        END IF;
        IF CurrentGroup = NILOBJ
                CurrentGroup := ASK GasLoadingDock TO FindHighPri();
        END IF;
        IF CurrentGroup = NILOBJ
                CurrentGroup := ASK PaxLoadingDock TO FindHighPri();
        END IF;
END CASE;

IF CurrentGroup <> NILOBJ
        Destination := ASK CurrentGroup Destination;
        LoadCargo(Destination, Transporter, Load);
        LoadGas(Destination, Transporter, Load);
        LoadPax(Destination, Transporter, Load);
END IF;

IF Load.numberIn > 0

        ASK Transporter TO LoadOut(Load);
        TELL Transporter TO GoTo(Destination);
ELSE
        ASK TransporterManager TO ReceiveAvailableTransporter(Transporter);
        ASK BerthsQ TO RemoveThis(Transporter);
        ASK ParkedQ TO Add(Transporter);
END IF;
DISPOSE(Load);

END METHOD;
```

```
{---------------------------------------------------------------------------}
ASK METHOD LoadCargo(INOUT Destination : BaseObj; INOUT Transporter :
        TransporterObj; INOUT Load : LoadQObj);
{---------------------------------------------------------------------------}

{BUILDS LOAD OF BREAKBULK CARGO}

VAR

i, j, k, Stop1, Stop2 : INTEGER;
Group, OverSizeGroup, CurrentGroup : CargoGroupObj;
CurrentShipment : ShipmentObj;

BEGIN


OUTPUT("IN LoadCargo - ", Class, " - ", Owner.Name);



{if transporter is oversize find hi pris in regular and oversize cargo. if overs

IF Transporter.OverSize

        {Find Destination Groups}

        Stop1 := ASK OverSizeLoadingDock numberIn;
        OverSizeGroup := NILOBJ;
        FOR i := 1 TO Stop1
                CurrentGroup := ASK OverSizeLoadingDock TO Remove;
                ASK OverSizeLoadingDock TO Add(CurrentGroup);
                IF CurrentGroup.Destination = Destination
                        OverSizeGroup := CurrentGroup;
                END IF;
        END FOR;
        Stop1 := ASK LoadingDock numberIn;
        Group := NILOBJ;
        FOR i := 1 TO Stop1
                CurrentGroup := ASK LoadingDock TO Remove;
                ASK LoadingDock TO Add(CurrentGroup);
                IF CurrentGroup.Destination = Destination
                        Group := CurrentGroup;
                END IF;
        END FOR;

        {Load alternately by priority}

        FOR i := 1 TO 12
            IF OverSizeGroup <> NILOBJ
                Stop2 := ASK OverSizeGroup numberIn;
                FOR k := 1 TO Stop2
                        CurrentShipment := ASK OverSizeGroup TO Remove;
                        IF CurrentShipment.Item.Priority = i
                                ASK SELF TO LoadItem(CurrentShipment,
                                 Transporter, Load, CurrentGroup);
                        END IF;
                        ASK OverSizeGroup TO Add(CurrentShipment);
                        IF CurrentShipment.Item.OnHand < 1.0
                                ASK OverSizeGroup TO
                                        RemoveThis(CurrentShipment);
```

```
                            DISPOSE(CurrentShipment);
                      END IF;
                END FOR;
             END IF;
             IF Group <> NILOBJ
                Stop2 := ASK Group numberIn;
                FOR k := 1 TO Stop2
                      CurrentShipment := ASK Group TO Remove;
                      IF CurrentShipment.Item.Priority = i
                            ASK SELF TO LoadItem(CurrentShipment,
                            Transporter, Load, CurrentGroup);
                      END IF;
                      ASK Group TO Add(CurrentShipment);
                      IF CurrentShipment.Item.OnHand < 1.0
                            ASK Group TO RemoveThis(CurrentShipment);
                            DISPOSE(CurrentShipment);
                      END IF;
                END FOR;
             END IF;

          END FOR;
          IF Group <> NILOBJ
                IF Group.numberIn < 1
                      ASK LoadingDock TO RemoveThis(Group);
                      DISPOSE(Group);
                ELSE
                      ASK LoadingDock TO RemoveThis(Group);
                      ASK LoadingDock TO Add(Group);
                      ASK Group TO ResetTallys;
                      ASK Group TO OrderTransporter;
                END IF;
          END IF;
          IF OverSizeGroup <> NILOBJ
                IF OverSizeGroup.numberIn < 1
                      ASK OverSizeLoadingDock TO RemoveThis(OverSizeGroup);
                      DISPOSE(OverSizeGroup);
                ELSE
                      ASK OverSizeLoadingDock TO RemoveThis(OverSizeGroup);
                      ASK OverSizeLoadingDock TO Add(OverSizeGroup);
                      ASK OverSizeGroup TO ResetTallys;
                      ASK OverSizeGroup TO OrderTransporter;
                END IF;
          END IF;
    ELSE
          Stop1 := ASK LoadingDock numberIn;
          Group := NILOBJ;
          FOR i := 1 TO Stop1
                CurrentGroup := ASK LoadingDock TO Remove;
                ASK LoadingDock TO Add(CurrentGroup);
                IF CurrentGroup.Destination = Destination
                      Group := CurrentGroup;
                END IF;
          END FOR;
                      {Load The rest of the shipments}
          FOR i := 1 TO 12
                IF Group <> NILOBJ
                    Stop2 := ASK Group numberIn;
                    FOR k := 1 TO Stop2
                      CurrentShipment := ASK Group TO Remove;
                      IF CurrentShipment.Item.Priority = i
```

```
                                        ASK SELF TO LoadItem(CurrentShipment,
                                                 Transporter, Load, CurrentGroup);
                                END IF;
                                ASK Group TO Add(CurrentShipment);
                                IF CurrentShipment.Item.OnHand < 1.0
                                        ASK Group TO RemoveThis(CurrentShipment);
                                        DISPOSE(CurrentShipment);
                                END IF;
                        END FOR;
                END IF;
        END FOR;

        IF Group <> NILOBJ
                IF Group.numberIn < 1
                        ASK LoadingDock TO RemoveThis(Group);
                        DISPOSE(Group);
                ELSE
                        ASK LoadingDock TO RemoveThis(Group);
                        ASK LoadingDock TO Add(Group);
                        ASK Group TO ResetTallys;
                        ASK Group TO OrderTransporter;
                END IF;
        END IF;
END IF;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD LoadGas(INOUT Destination : BaseObj; INOUT Transporter :
         TransporterObj; INOUT Load : LoadQObj);
{---------------------------------------------------------------------------}
VAR

i, k, Stop : INTEGER;
Group, CurrentGroup : CargoGroupObj;
CurrentShipment : ShipmentObj;

BEGIN

{
OUTPUT("IN LoadGas - ", Class, " - ", Owner.Name);
}

{Find Destination Groups}

Stop := ASK GasLoadingDock numberIn;
Group := NILOBJ;
FOR i := 1 TO Stop
        CurrentGroup := ASK GasLoadingDock TO Remove;
        ASK GasLoadingDock TO Add(CurrentGroup);
        IF CurrentGroup.Destination = Destination
                Group := CurrentGroup;
        END IF;
END FOR;

{Load The liquid shipments}
IF Group <> NILOBJ
        FOR i := 1 TO 12
                Stop := ASK Group numberIn;
                FOR k := 1 TO Stop
```

```
                                CurrentShipment := ASK Group TO Remove;
                                ASK Group TO Add(CurrentShipment);
                                IF CurrentShipment.Item.Priority = i
                                        ASK SELF TO LoadItem(CurrentShipment,
                                                 Transporter, Load, CurrentGroup);
                                END IF;
                                IF CurrentShipment.Item.OnHand < 1.0
                                        ASK Group TO RemoveThis(CurrentShipment);
                                        DISPOSE(CurrentShipment);
                                END IF;
                        END FOR;
                END FOR;

{clean-up cargo group, dispose in necessary, place at back of the queue}

                IF Group.numberIn < 1
                        ASK GasLoadingDock TO RemoveThis(Group);
                        DISPOSE(Group);
                ELSE
                        ASK GasLoadingDock TO RemoveThis(Group);
                        ASK GasLoadingDock TO Add(Group);
                        ASK Group TO ResetTallys;
                        ASK Group TO OrderTransporter;
                END IF;
END IF;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD LoadPax(INOUT Destination : BaseObj; INOUT Transporter :
         TransporterObj; INOUT Load : LoadQObj);
{-----------------------------------------------------------------------}
VAR

i, k, Stop : INTEGER;
Group, CurrentGroup : CargoGroupObj;
CurrentShipment : ShipmentObj;

BEGIN

{
OUTPUT("IN LoadPax - ", Class, " - ", Owner.Name);
}

{Find Destination Groups}

Stop := ASK PaxLoadingDock numberIn;
Group := NILOBJ;
FOR i := 1 TO Stop
        CurrentGroup := ASK PaxLoadingDock TO Remove;
        ASK PaxLoadingDock TO Add(CurrentGroup);
        IF CurrentGroup.Destination = Destination
                Group := CurrentGroup;
        END IF;
END FOR;

{Load The pax shipments}
IF Group <> NILOBJ
        FOR i := 1 TO 12
                Stop := ASK Group numberIn;
```

```
                    FOR k := 1 TO Stop
                            CurrentShipment := ASK Group TO Remove;
                            ASK Group TO Add(CurrentShipment);
                            IF CurrentShipment.Item.Priority = i
                                    ASK SELF TO LoadItem(CurrentShipment,
                                             Transporter, Load, CurrentGroup);
                            END IF;
                            IF CurrentShipment.Item.OnHand < 1.0
                                    ASK Group TO RemoveThis(CurrentShipment);
                                    DISPOSE(CurrentShipment);
                            END IF;
                    END FOR;
            END FOR;

    {clean-up cargo group, dispose in necessary, place at back of the queue}

            IF Group.numberIn < 1
                    ASK PaxLoadingDock TO RemoveThis(Group);
                    DISPOSE(Group);
            ELSE
                    ASK PaxLoadingDock TO RemoveThis(Group);
                    ASK PaxLoadingDock TO Add(Group);
                    ASK Group TO ResetTallys;
                    ASK Group TO OrderTransporter;
            END IF;
    END IF;

    END METHOD;

    {-------------------------------------------------------------------------}
        ASK METHOD LoadItem(INOUT Shipment : ShipmentObj; IN Transporter :
    TransporterObj; INOUT Load : LoadQObj; INOUT CurrentGroup : CargoGroupObj); {---

    {LOADS AN INDIVIDUAL SHIPMENT TO LOAD UNDER CONSTRUCTION, CALLED BY LOAD}

    VAR

    TotalItemCube : REAL;
    TotalItemWeight : REAL;
    TotalItemArea : REAL;
    Item : CommodityObj;
    ItemCube : REAL;
    ItemWeight : REAL;
    ItemArea : REAL;
    WeightDifference : REAL;
    CubeDifference : REAL;
    AreaDifference : REAL;
    CubeNumToTake : REAL;
    WeightNumToTake : REAL;
    AreaNumToTake : REAL;
    NumToTake : REAL;
    NewShipment : ShipmentObj;
    PaxNumToTake : REAL;
    Destination : BaseObj;
    Route : BaseQObj;

    BEGIN


    OUTPUT("IN LoadItem - ", Shipment.Item.Name, " - ", Owner.Name);
```

```
Item := Shipment.Item;
ItemCube := Item.Length * Item.Width * Item.Height/1728.00 {cu in. per cu ft.};
TotalItemCube := Item.OnHand * ItemCube;
ItemWeight := Item.Weight;
TotalItemWeight := Item.OnHand * ItemWeight;
ItemArea:= Item.Length * Item.Width / 144.00 {sq in. per sq. ft.};
TotalItemArea := ItemArea * Item.OnHand;

CubeDifference  := Transporter.MaxCargoCube - (Load.TotalCube);
IF ItemCube <> 0.0
        CubeNumToTake := FLOAT(TRUNC(CubeDifference/ItemCube));
ELSE
        CubeNumToTake := Item.OnHand;
END IF;

AreaDifference  := Transporter.MaxCargoArea - (Load.TotalArea);
IF ItemArea <> 0.0
        AreaNumToTake := FLOAT(TRUNC(AreaDifference/ItemArea));
ELSE
        AreaNumToTake := Item.OnHand;
END IF;

WeightDifference := Transporter.MaxCargoWeight  -  (Load.TotalWeight);
IF ItemWeight <> 0.0
        WeightNumToTake :=  FLOAT(TRUNC(WeightDifference/ItemWeight));
ELSE
        WeightNumToTake := Item.OnHand;
END IF;

{
OUTPUT("                                       CubeNumToTake = ", CubeNumToTake);
OUTPUT("                                       WeightNumToTake = ", WeightNumToTake);
OUTPUT("                                       AreaNumToTake = ", AreaNumToTake);
}

CASE Item.Class
WHEN Fuel:
        IF (Transporter.SubClass <> General) OR (Transporter.SubClass <> Pax)

{
                OUTPUT("TRANSPORTER.MAXGAS = ", Transporter.MaxGas);
                OUTPUT("LOAD.MAXGAS = ", Load.TotalGas);
}

                IF ((Load.TotalGas + Item.OnHand) <= Transporter.MaxGas)
                        NumToTake := Item.OnHand;
                ELSE
                        NumToTake :=  Transporter.MaxGas - Load.TotalGas;
                END IF;
        END IF;
WHEN Personnel:

{
        OUTPUT("LOAD.TOTALPAX = ", Load.TotalPax);
        OUTPUT("TRANSPORTER.MAXPAX = ", Transporter.MaxPax);
}

        IF ((Load.TotalPax + Item.OnHand) <= Transporter.MaxPax) AND
```

```
             ((Load.TotalWeight + TotalItemWeight) <= Transporter.MaxCargoWeight);
                     NumToTake := Item.OnHand;
            ELSE
                     NumToTake := Transporter.MaxPax - Load.TotalPax;
            END IF;

    WHEN Major:

            IF (Transporter.SubClass <> Liquid) OR (Transporter.SubClass <> Pax)

    {
                     OUTPUT("LOAD.TOTALAREA = ", Load.TotalArea);
                     OUTPUT("TRANSPORTER.MAXAREA = ",Transporter.MaxCargoArea);
                     OUTPUT("LOAD.TOTALCUBE = ", Load.TotalCube);
                     OUTPUT("TRANSPORTER.MAXCUBE = ", Transporter.MaxCargoCube);
                     OUTPUT("LOAD.TOTALWEIGHT = ", Load.TotalWeight);
                     OUTPUT("TRANSPORTER.MAXWEIGHT = ", Transporter.MaxCargoWeight);
    }

                     IF ((Load.TotalCube + TotalItemCube) <=
                      Transporter.MaxCargoCube) AND ((Load.TotalWeight +
                      TotalItemWeight) <= Transporter.MaxCargoWeight) AND
                      ((Load.TotalArea + TotalItemArea) <= Transporter.MaxCargoArea)
                             NumToTake := Item.OnHand;
                     ELSE
                             NumToTake := MINOF(WeightNumToTake, CubeNumToTake,
                                      AreaNumToTake);
                     END IF;
            END IF;
    OTHERWISE
            IF (Transporter.SubClass <> Liquid) OR (Transporter.SubClass <> Pax)

    {
                     OUTPUT("LOAD.TOTALCUBE = ", Load.TotalCube);
                     OUTPUT("TRANSPORTER.MAXCUBE = ", Transporter.MaxCargoCube);
                     OUTPUT("LOAD.TOTALWEIGHT = ", Load.TotalWeight);
                     OUTPUT("TRANSPORTER.MAXWEIGHT = ", Transporter.MaxCargoWeight);
    }

                     IF ((Load.TotalCube + TotalItemCube) <=
                      Transporter.MaxCargoCube) AND ((Load.TotalWeight +
                      TotalItemWeight) <= Transporter.MaxCargoWeight)
                             NumToTake := Item.OnHand;
                     ELSE
                             NumToTake := MINOF(WeightNumToTake,CubeNumToTake);
                     END IF;
            END IF;
    END CASE;

    {
    OUTPUT("                                 NUMTOTAKE = ", NumToTake);
    }

    IF NumToTake >= 1.0
            NEW(NewShipment);
            Destination := ASK Shipment Destination;
            ASK NewShipment TO SetDestination(Destination);
            ASK NewShipment TO SetRDD(Shipment.RDD);
            Route := ASK Shipment Route;
            ASK NewShipment TO SetRoute(Route);
```

```
        ASK NewShipment TO SetItem(Shipment.Item);
        ASK Shipment.Item TO SubtractOnHand(NumToTake);
        ASK NewShipment.Item TO SetOnHand(NumToTake);
        ASK Load TO Add(NewShipment);
END IF;

END METHOD;

{----------------------------------------------------------------------------}
     ASK METHOD ObjInit; {-------------------------------------------------------
VAR

BEGIN

NEW(BerthsQ);
NEW(ArrivalsQ);
NEW(ParkedQ);
NEW(LoadingDock);
NEW(OverSizeLoadingDock);
NEW(PaxLoadingDock);
NEW(GasLoadingDock);
NEW(Network);

END METHOD;

{----------------------------------------------------------------------------}
     ASK METHOD ObjTerminate; {--------------------------------------------------
VAR

BEGIN

ASK Network TO Empty;
DISPOSE(Network);
ASK BerthsQ TO Empty;
DISPOSE(BerthsQ);
ASK ParkedQ TO Empty;
DISPOSE(ParkedQ);
ASK ArrivalsQ TO Empty;
DISPOSE(ArrivalsQ);
DISPOSE(LoadingDock);
DISPOSE(OverSizeLoadingDock);
DISPOSE(PaxLoadingDock);
DISPOSE(GasLoadingDock);

END METHOD;

END OBJECT;

END MODULE.
```

```
DEFINITION MODULE SOUTPUT;

{SOUTPUT for Special OUTPUT.  Paused the OUTPUT of a large list if it exceed one

{Import statements}

{FROM IMPORT}

{Type Declarations}

VAR

PROCEDURE SOUTPUT(IN string : STRING; INOUT i : INTEGER);

END MODULE.
```

```
IMPLEMENTATION MODULE SOUTPUT;

{Comments}

{Import statements}


FROM CRTMod IMPORT ClearScreen;
FROM IOMod IMPORT ReadKey;

{Definitions}

{
{=========================================================================}
OBJECT ObjName
{=========================================================================}

     {METHODS}

{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}
VAR

BEGIN

END METHOD;

END OBJECT;
}
{-------------------------------------------------------------------------}
PROCEDURE SOUTPUT(IN string : STRING; INOUT i : INTEGER);
{-------------------------------------------------------------------------}
VAR

CHR : CHAR;

BEGIN

IF i < 32
        OUTPUT(string);
        i := i +1;
ELSE
        OUTPUT("Press any key to continue.");
        CHR := ReadKey();
        OUTPUT(string);
        i := 1;
END IF;


END PROCEDURE;

END MODULE.
```

```
DEFINITION MODULE ScenEd;

{The ScenarioEditor is a refinement of the Builder, it iteractively creates
 objects and stores them in datafiles to be read by the Builder during
execution}

{Import statements}

FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM CommodQ IMPORT ALL CommodityClassType,
                    CommodityObj,
                    CommodityQObj;
FROM Trnsprt IMPORT TransporterObj,
                    TransporterQObj,
                    ALL TransporterClassType;
FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM Port IMPORT PortObj;
FROM Unit IMPORT UnitObj,
                 UnitQObj;

FROM Shpmnt IMPORT ShipmentObj;
FROM Scene IMPORT ScenarioQObj;
FROM SubUnit IMPORT SubUnitObj,
                    SubUnitQObj;
FROM Distant IMPORT PositionRecType;
FROM Builder IMPORT BuilderObj;
FROM IOMod IMPORT StreamObj,
                  ALL FileUseType,
                  ReadKey;


{
FROM IMPORT ;
}

{Type Declarations}

TYPE

{===========================================================================}
ScenarioEditorObj  = OBJECT(BuilderObj);
{===========================================================================}

{FIELDS}

ScenarioList : ScenarioQObj;

Planes : SubUnitQObj;
Ships : SubUnitQObj;
Tanks : SubUnitQObj;

{METHODS}

{X} ASK METHOD CreateBase;
{X} ASK METHOD CreateCommodity;
{X} ASK METHOD CreateUnit;
{X} ASK METHOD CreateTransporter;
{x} ASK METHOD CreateScenario;
```

```
{x} ASK METHOD CreatePrepo;

{X} ASK METHOD SaveMasterTransporterFile;
{X} ASK METHOD SaveTransporterDataFile(IN Transporter : TransporterObj);
{x} ASK METHOD SaveMasterBaseFile;
{x} ASK METHOD SaveBaseDataFile(IN base : BaseObj);
{x} ASK METHOD SaveMasterUnitFile;
{x} ASK METHOD SaveMasterSubUnitFile;
{x} ASK METHOD SaveUnitDataFile(IN unit : UnitObj);
{x} ASK METHOD SaveMasterCommodityFile;
{x} ASK METHOD SaveScenarioMasterFile;
{x} ASK METHOD SavePrepoMasterFile;
{x} ASK METHOD SavePrepoDataFile(IN Name : STRING; IN transporter :
         TransporterObj; IN shipment : ShipmentObj; IN cargo : CommodityQObj);


{x} ASK METHOD InputHasPorts(INOUT base : BaseObj);
{x} ASK METHOD InputPort() : PortObj;
{x} ASK METHOD InputTransporters(INOUT base : BaseObj);
{x} ASK METHOD InputCommodities(INOUT base : BaseObj);
{x} ASK METHOD InputSubUnits(INOUT unit : UnitObj; IN SubUnitQ : SubUnitQObj);
{x} ASK METHOD CountTransporters(IN Port : PortObj; IN file : StreamObj);
{x} ASK METHOD CountTransporterTypes(IN Port : PortObj; IN file : StreamObj) :
                                                                      INTEGER;
{x} ASK METHOD ReadBaseMasterFile;
{x} ASK METHOD ReadUnitMasterFile;
{x} ASK METHOD ReadSubUnitMasterFile;
{x} ASK METHOD ReadTransporterMasterFile;
{x} ASK METHOD ReadCommodityMasterFile;

{} ASK METHOD ReadPrepoMasterFile;
{x} ASK METHOD ReadScenarioMasterFile;

OVERRIDE

{x} ASK METHOD ObjInit;
{x} ASK METHOD ObjTerminate;



END OBJECT;

VAR
ScenarioEditor : ScenarioEditorObj;
END MODULE.
```

```
IMPLEMENTATION MODULE ScenEd;

{Comments}

{Import statements}
FROM Trash IMPORT GarbageDisposal;
FROM IOMod IMPORT StreamObj,
                  ALL FileUseType,
                  ReadKey;
FROM MyQueue IMPORT NamedObj,
                    MyQueueObj;
FROM Base IMPORT BaseObj,
                 BaseQObj,
                 ALL BaseGroupType,
                 ALL BaseSubGroupType;
FROM Port IMPORT PortObj;
FROM Unit IMPORT UnitObj,
                 UnitQObj,
                 ALL UnitClassType,
                 ALL CombatIntensityType;
FROM Shpmnt IMPORT ShipmentObj;
FROM SubUnit IMPORT SubUnitObj,
                    SubUnitQObj;
FROM CommodQ IMPORT ALL CommodityClassType,
                    CommodityObj,
                    CommodityQObj;

FROM Trnsprt IMPORT TransporterObj,
                    TransporterQObj,
                    ALL TransporterClassType,
                    ALL TransporterSubClassType;
FROM Distant IMPORT PositionRecType,
                    InputPosition,
                    ALL MinType,
                    ALL LatDegType,
                    ALL LatDirType,
                    ALL LongDegType,
                    ALL LongDirType;

FROM Port IMPORT PortObj;
FROM Scene IMPORT ScenarioObj;

FROM CRTMod IMPORT ClearScreen;
FROM SOUTPUT IMPORT SOUTPUT;
{
FROM IMPORT ;
}

{Definitions}

{===========================================================================}
OBJECT ScenarioEditorObj;
{===========================================================================}

     {METHODS}

{---------------------------------------------------------------------------}
ASK METHOD CreateBase;
{---------------------------------------------------------------------------}
```

```
{INTERACTIVLY BUILDS A BASE AND ADDS IT TO THE SCENARIO EDITOR BASEQ}

VAR

base : BaseObj;
commodity : CommodityObj;
transporterRec : TransporterObj;
string : STRING;
position : PositionRecType;
port : PortObj;
CHR : CHAR;

BEGIN

NEW(base);
OUTPUT(" Creating Base - Scenario Editor");
ClearScreen;
OUTPUT("                              CREATING A NEW BASE");
ASK base TO InputName;
ASK base TO InputGroup;
ASK base TO InputSubGroup;

position := InputPosition();
ASK base TO SetPosition(position);

ASK SELF TO InputHasPorts(base);

IF base.HasAirPort
        OUTPUT("     Building Airport.");
        port := ASK SELF TO InputPort;
        ASK base TO SetAirPort(port);
        ASK base.AirPort TO SetClass("Aircraft");
        ASK base.AirPort TO SetOwner(base);
        DISPOSE(port);
END IF;

IF base.HasSeaPort
        OUTPUT("     Building Seaport.");
        port := ASK SELF TO InputPort;
        ASK base TO SetSeaPort(port);
        ASK base.SeaPort TO SetClass("Ship");
        ASK base.SeaPort TO SetOwner(base);
        DISPOSE(port);
END IF;

IF base.HasRail
        OUTPUT("     Building Railyard.");
        port := ASK SELF TO InputPort;
        ASK base TO SetRailYard(port);
        ASK base.RailYard TO SetClass("Rail");
        ASK base.RailYard TO SetOwner(base);
        DISPOSE(port);
END IF;

IF base.HasTruckStop
        OUTPUT("     Building Truck Stop.");
        port := ASK SELF TO InputPort;
        ASK base TO SetTruckStop(port);
        ASK base.TruckStop TO SetClass("Truck");
        ASK base.TruckStop TO SetOwner(base);
        DISPOSE(port);
END IF;

ASK base TO Display;

LOOP
        OUTPUT("     Deploy Transporters to Ports? (Y)");
        CHR := ReadKey();
        IF (CHR = "Y") OR (CHR = "y")
                ASK SELF TO InputTransporters(base);
        ELSIF (CHR = "N") OR (CHR = "n")
                EXIT
        END IF;

END LOOP;

LOOP
        OUTPUT("     Add Commodities to Inventory? (Y)");
        CHR := ReadKey();
        IF (CHR = "Y") OR (CHR = "y")
                ASK SELF TO InputCommodities(base);
        ELSIF (CHR = "N") OR (CHR = "n")
                EXIT
        END IF;

END LOOP;

LOOP
        ASK base TO Display;
        OUTPUT("     Save Base (Y or N)");
        CHR := ReadKey();
        IF  (CHR = "Y") OR (CHR = "y")
                ASK BaseQ TO Add(base);
                ASK OnlyBaseQ TO Add(base);
                EXIT;
        ELSIF  (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;
END LOOP;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD CreateCommodity;
{---------------------------------------------------------------------}

{INTERACTIVEY BUILDS A COMMODITY AND ADDS IT TO THE SCENARIO EDITOR COMMODITQ}

VAR

commodity : CommodityObj;
string : STRING;
real : REAL;
j, integer : INTEGER;
CHR : CHAR;

BEGIN
ClearScreen;
NEW(commodity);
```

```
ASK commodity TO InputName;

LOOP
        OUTPUT("     Input Commodity Class:  ");
        OUTPUT("     (F)uel, FF(V), (A)mmo, (S)pares, (P)ersonnel, (M)edical, Ma
        CHR := ReadKey();
        IF (CHR = "F") OR (CHR = "f")
                ASK commodity TO SetClass("Fuel");
                EXIT;
        ELSIF  (CHR = "V") OR (CHR = "v")
                ASK commodity TO SetClass("FFV");
                EXIT;
        ELSIF  (CHR = "A") OR (CHR = "a")
                ASK commodity TO SetClass("Ammo");
                EXIT;
        ELSIF  (CHR = "S") OR (CHR = "s")
                ASK commodity TO SetClass("Spares");
                EXIT;
        ELSIF  (CHR = "P") OR (CHR = "p")
                ASK commodity TO SetClass("Personnel");
                EXIT;
        ELSIF  (CHR = "M") OR (CHR = "m")
                ASK commodity TO SetClass("Medical");
                EXIT;
        ELSIF  (CHR = "R") OR (CHR = "r")
                ASK commodity TO SetClass("Major");
                EXIT;
        ELSIF  (CHR = "O") OR (CHR = "o")
                ASK commodity TO SetClass("Other");
                EXIT;
        END IF;
END LOOP;

OUTPUT("     Input Commodity Production Rate (REAL numbers)");
INPUT(real);
ASK commodity TO SetProduceAt(real);

OUTPUT("     Input Commodity Length (REAL inches)");
INPUT(real);
ASK commodity TO SetLength(real);

OUTPUT("     Input Commodity Width (REAL inches)");
INPUT(real);
ASK commodity TO SetWidth(real);

OUTPUT("     Input Commodity Height (REAL inches)");
INPUT(real);
ASK commodity TO SetHeight(real);

OUTPUT("     Input Commodity Weight (REAL inches)");
INPUT(real);
ASK commodity TO SetWeight(real);

LOOP
        OUTPUT("     Input Commodity Priority (1-12)");
        INPUT(integer);
        IF (integer > 0) AND (integer < 13)
                EXIT;
        END IF;
END LOOP;
ASK commodity TO SetNormalPriority(integer);

LOOP
        OUTPUT("     Input Commodity Emergency Priority (1-12)");
        INPUT(integer);
        IF (integer > 0) AND (integer < 13)
                EXIT;
        END IF;
END LOOP;
ASK commodity TO SetEmerPriority(integer);

IF (commodity.Length > 1090.0) OR (commodity.Width > 117.0) OR
                                                (commodity.Height > 105.0)
        ASK commodity TO SetOverSize("TRUE");
ELSE
        ASK commodity TO SetOverSize("FALSE");
END IF;


LOOP
        ClearScreen;
        OUTPUT(" ");
        SOUTPUT(  "     Name             Class       Dim. (inches)    Weight  Pro
OUTPUT("=========================================================================
        ASK commodity TO Display(j);
        OUTPUT(" ");
        OUTPUT("     Save Commodity (Y or N)?");
        CHR := ReadKey();
        IF  (CHR = "Y") OR (CHR = "y")
                ASK CommodityQ TO Add(commodity);
                EXIT;
        ELSIF  (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;
END LOOP;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD CreateTransporter;
{---------------------------------------------------------------------------}

{INTERACTIVEY BUILDS A TRANSPORTER AND ADDS IT TO THE SCENARIO EDITOR
TRANSPORTERQ}

VAR

transporter : TransporterObj;
string : STRING;
real : REAL;
j, integer : INTEGER;
CHR : CHAR;

BEGIN

NEW(transporter);
ASK transporter TO SetVehicleID(1);

ASK transporter TO InputName;
ASK transporter TO InputClass;
ASK transporter TO InputSubClass;

OUTPUT(" ");
OUTPUT("     Input Transporter Max Speed (REAL number)");
INPUT(real);
ASK transporter TO SetMaxSpeed(real);

OUTPUT("     Input Transporter Max Range (REAL nautical miles)");
INPUT(real);
ASK transporter TO SetMaxRange(real);

OUTPUT("     Input Transporter Length (REAL feet)");
INPUT(real);
ASK transporter TO SetLength(real);

OUTPUT("     Input Transporter Width (REAL feet)");
INPUT(real);
ASK transporter TO SetWidth(real);

OUTPUT("     Input Transporter Cargo Area (REAL square feet)");
INPUT(real);
ASK transporter TO SetMaxCargoArea(real);

OUTPUT("     Input Transporter Cargo Cube (REAL cubic feet)");
INPUT(real);
ASK transporter TO SetMaxCargoCube(real);

OUTPUT("     Input Transporter Max Cargo Length (REAL inches)");
INPUT(real);
ASK transporter TO SetMaxCargoLength(real);

OUTPUT("     Input Transporter Max Cargo Width (REAL inches)");
INPUT(real);
ASK transporter TO SetMaxCargoWidth(real);

OUTPUT("     Input Transporter Max Cargo Height (REAL inches)");
INPUT(real);
ASK transporter TO SetMaxCargoHeight(real);

OUTPUT("     Input Transporter Max Cargo Weight (REAL pounds)");
INPUT(real);
ASK transporter TO SetMaxCargoWeight(real);

OUTPUT("     Input Transporter Max Passenger Capacity (persons)");
INPUT(real);
ASK transporter TO SetMaxPax(real);

OUTPUT("     Input Transporter Max Liquid Capacity (REAL Barrels)");
INPUT(real);
ASK transporter TO SetMaxGas(real);


IF (transporter.MaxCargoLength > 1090.0) OR (transporter.MaxCargoWidth > 117.0)
                                         OR (transporter.MaxCargoHeight > 105.0)
        ASK transporter TO SetOverSize("TRUE");
ELSE
        ASK transporter TO SetOverSize("FALSE");
END IF;
```

```
LOOP
        ClearScreen;
        OUTPUT(" ");
        ASK transporter TO Display;
        OUTPUT(" ");
        OUTPUT("     Save Transporter (Y or N)?");
        CHR := ReadKey();
        IF  (CHR = "Y") OR (CHR = "y")
                ASK TransporterQ TO Add(transporter);
                EXIT;
        ELSIF  (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;
END LOOP;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD CreateUnit;
{-----------------------------------------------------------------------}

{INTERACTIVEY BUILDS A UNIT AND ADDS IT TO THE SCENARIO EDITOR UNITQ}

VAR

unit : UnitObj;
commodity : CommodityObj;
transporterRec : TransporterObj;
string : STRING;
position : PositionRecType;
port : PortObj;
CHR : CHAR;
SubUnitQ : SubUnitQObj;
j : INTEGER;

BEGIN

NEW(unit);
OUTPUT("creating unit");
ClearScreen;
OUTPUT("                              CREATING A NEW UNIT");
ASK unit TO InputName;
ASK unit TO InputDelayUntil;
LOOP
OUTPUT("     What type of unit? (A)ir, (L)and, (S)ea.");
CHR := ReadKey();
IF (CHR = "A") OR (CHR = "a")
        ASK unit TO SetClass("Air");
        SubUnitQ := Planes;
        EXIT;
ELSIF (CHR = "L") OR (CHR = "l");
        ASK unit TO SetClass("Land");
        SubUnitQ := Tanks;
        EXIT;
ELSIF (CHR = "S") OR (CHR = "s");
        ASK unit TO SetClass("Sea");
        SubUnitQ := Ships;
        EXIT;
```

```
END IF;
END LOOP;

position := InputPosition();
ASK unit TO SetPosition(position);

ASK SELF TO InputHasPorts(unit);

IF unit.HasAirPort
        ASK unit TO Display;
        OUTPUT("     Building Airport.");
        port := ASK SELF TO InputPort;
        ASK unit TO SetAirPort(port);
        ASK unit.AirPort TO SetClass("Aircraft");
        ASK unit.AirPort TO SetOwner(unit);
        DISPOSE(port);
END IF;

IF unit.HasSeaPort
        ASK unit TO Display;
        OUTPUT("     Building Seaport.");
        port := ASK SELF TO InputPort;
        ASK unit TO SetSeaPort(port);
        ASK unit.SeaPort TO SetClass("Ship");
        ASK unit.SeaPort TO SetOwner(unit);
        DISPOSE(port);
END IF;

IF unit.HasRail
        ASK unit TO Display;
        OUTPUT("     Building Railyard.");
        port := ASK SELF TO InputPort;
        ASK unit TO SetRailYard(port);
        ASK unit.RailYard TO SetClass("Rail");
        ASK unit.RailYard TO SetOwner(unit);
        DISPOSE(port);
END IF;

IF unit.HasTruckStop
        ASK unit TO Display;
        OUTPUT("     Building Truck Stop.");
        port := ASK SELF TO InputPort;
        ASK unit TO SetTruckStop(port);
        ASK unit.TruckStop TO SetClass("Truck");
        ASK unit.TruckStop TO SetOwner(unit);
        DISPOSE(port);
END IF;

LOOP
        OUTPUT("     Add Transporters? (Y)");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;
        ASK SELF TO InputTransporters(unit);
        OUTPUT("     Return (Y)");
        CHR := ReadKey();
        IF (CHR = "Y") OR (CHR = "y")
                EXIT
        END IF;
END LOOP;

LOOP
        ClearScreen;
        j := 0;
        ASK unit.Inventory TO Display(j);
        CASE unit.Class
        WHEN Air:
                OUTPUT("     Add Aircraft? (Y)");
                CHR := ReadKey();
                IF (CHR <> "N") AND (CHR <> "n")
                        ASK SELF TO InputSubUnits(unit,Planes);
                END IF;
        WHEN Sea:
                OUTPUT("     Add Ships? (Y)");
                CHR := ReadKey();
                IF (CHR <> "N") AND (CHR <> "n")
                        ASK SELF TO InputSubUnits(unit,Ships);
                END IF;
                OUTPUT("     Add Aircraft? (Y)");
                CHR := ReadKey();
                IF (CHR <> "N") AND (CHR <> "n")
                        ASK SELF TO InputSubUnits(unit,Planes);
                END IF;
        WHEN Land:
                OUTPUT("     Add Land Units? (Y)");
                CHR := ReadKey();
                IF (CHR <> "N") AND (CHR <> "n")
                        ASK SELF TO InputSubUnits(unit,Tanks);
                END IF;
                OUTPUT("     Add Aircraft? (Y)");
                CHR := ReadKey();
                IF (CHR <> "N") AND (CHR <> "n")
                        ASK SELF TO InputSubUnits(unit,Planes);
                END IF;
        OTHERWISE
        END CASE;
        OUTPUT("     Finished adding SubUnits? (Y)");
        CHR := ReadKey();
        IF (CHR <> "N") OR (CHR <> "n")
                EXIT;
        END IF;


END LOOP;

LOOP
        ClearScreen;
        j := 0;
        ASK unit.Inventory TO Display(j);
        OUTPUT("     Input other Commodities? (Y)");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;
        ASK SELF TO InputCommodities(unit);
END LOOP;

LOOP
```

```
        ASK unit TO Modify(ScenarioEditor);
        OUTPUT("     Save Unit (Y or N)");
        CHR := ReadKey();
        IF  (CHR = "Y") OR (CHR = "y")
                ASK UnitQ TO Add(unit);
                EXIT;
        ELSIF  (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;
END LOOP;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD CreateScenario;
{---------------------------------------------------------------------------}

{INTERACTIVEY BUILDS A SCENARIO AND ADDS IT TO THE SCENARIO EDITOR SCENARIOLIST}

VAR

Scenario : ScenarioObj;

BEGIN

ClearScreen;
NEW(Scenario);
{ASK Scenario.BaseQ TO Add(Supply);}
OUTPUT("                              BUILDING NEW SCENARIO" );
ASK Scenario TO InputName;
ASK Scenario TO PickBases;
ASK Scenario TO PickUnits;
ASK Scenario TO PickPrepos;
ASK Scenario TO CreateBaseFile;
ASK Scenario TO CreateUnitFile;
ASK Scenario TO CreatePrepoFile;
ASK Scenario TO CreateBaseLinkFile;

ASK Scenario TO CreateRailLinkFile;
ASK Scenario TO CreateTruckLinkFile;
ASK Scenario TO CreateScenarioFile;
ASK ScenarioList TO Add(Scenario);

END METHOD;


{---------------------------------------------------------------------------}
ASK METHOD CreatePrepo;
{---------------------------------------------------------------------------}

{INTERACTIVEY BUILDS A PREPO AND ADDS IT TO THE SCENARIO EDITOR PREPOQ}

VAR

NewTransporter, transporter : TransporterObj;
string, Name : STRING;
real : REAL;
j, i, k, integer : INTEGER;
CHR : CHAR;
```

```
base : BaseObj;
shipment : ShipmentObj;
commodity, NewCommodity : CommodityObj;
cargo : CommodityQObj;
Prepo : NamedObj;
position : PositionRecType;

BEGIN
NEW(Prepo);
NEW(cargo);
OUTPUT("     Input prepositioned transporter name");
INPUT(Name);
ASK Prepo TO SetName(Name);

ClearScreen;
LOOP
        j:=0;
        SOUTPUT(" ",j);
        SOUTPUT(" ",j);
        ASK TransporterQ TO Display(j);
        SOUTPUT("============================================================="
                 + "===================",j);
        OUTPUT("");
        OUTPUT("     Input Transporter Name");
        INPUT(string);
        transporter := ASK TransporterQ TO FindByName(string);
        IF transporter <> NILOBJ
                NewTransporter := CLONE(transporter);
                EXIT;
        END IF;
END LOOP;

OUTPUT("     Entering transporter position data");
position := InputPosition();
ASK NewTransporter TO SetPosition(position);
NEW(shipment);
DISPOSE(position);
LOOP
        ClearScreen;
        j := 0;
        ASK BaseQ TO Display(j);
        SOUTPUT("============================================================="
                 + "===================",j);
        OUTPUT("");
        OUTPUT("     Input the ultimate destination of the prepositioned"
                + " cargo.");
        INPUT(string);
        base := ASK BaseQ TO FindByName(string);
        IF base <> NILOBJ
                ASK shipment TO SetDestination(base);
                EXIT;
        END IF;
END LOOP;

LOOP
        OUTPUT("");
        OUTPUT("Current route of cargo is:");
        integer := ASK shipment.Route numberIn;
        FOR i := 1 TO integer
                base := ASK shipment.Route TO Remove;
                ASK shipment.Route TO Add(base);
                OUTPUT("      ", i, ".  ", base.Name);
        END FOR;
        OUTPUT("      DESTINATION:  ",  shipment.Destination.Name);
        OUTPUT("");
        OUTPUT("      Add an intermediate destination? (Y or N)");

        CHR := ReadKey();
        IF (CHR = "Y") OR (CHR = "y")
                LOOP
                        ClearScreen;
                        j := 0;
                        ASK BaseQ TO Display(j);
                        SOUTPUT("=============================================="
                         + "==================================",j);
                        OUTPUT("");
                        OUTPUT("      Input intermediate destination.");
                        INPUT(string);
                                        base := ASK BaseQ TO FindByName(string);
                        IF base <> NILOBJ
                                ASK shipment.Route TO Add(base);
                                EXIT;
                        END IF;
                END LOOP;
        ELSIF (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;

END LOOP;
ASK shipment.Route TO Add(shipment.Destination);
LOOP
        ClearScreen;
        j := 0;
        ASK CommodityQ TO Display(j);
        OUTPUT("");
        OUTPUT("      Add a commodity to the cargo list? (Y or N)");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n")
                EXIT;
        ELSIF (CHR = "Y") OR (CHR = "y")
                LOOP
                        OUTPUT("      Input commodity name.");
                        INPUT(string);
                        commodity := ASK CommodityQ TO FindByName(string);
                        IF commodity <> NILOBJ
                                NewCommodity := CLONE(commodity);
                                ASK cargo TO Add(NewCommodity);
                                OUTPUT("      How much do you want on hand?");
                                INPUT(real);
                                ASK NewCommodity TO SetOnHand(real);
                                EXIT;
                        END IF;
                END LOOP;
        END IF;
END LOOP;

ASK PrepoQ TO Add(Prepo);
SavePrepoDataFile(Name, NewTransporter, shipment, cargo);
DISPOSE(cargo);
DISPOSE(NewTransporter);
DISPOSE(shipment);

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD SavePrepoMasterFile;
{----------------------------------------------------------------------}

{BUILDS A PREPO MASTER DATAFILE}

VAR

object : NamedObj;
i, numItems : INTEGER;
file : StreamObj;
string, string2 : STRING;

BEGIN

NEW(file);
ASK file TO Open("Prepo.mst", Output);
numItems := ASK PrepoQ numberIn;
ASK file TO WriteString("NumberOfFiles: "+ INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        object := ASK PrepoQ TO Remove;
        ASK PrepoQ TO Add(object);
        string := ASK object Name;
{       string2 := SUBSTR(1,8,string);
        string := string2 + ".dat" ;}
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD ReadPrepoMasterFile;
{----------------------------------------------------------------------}

{READS A PREPO MASTER DATAFILE}

CONST

MasterFile = "Prepo.mst";

VAR

File : StreamObj;
prepo : NamedObj;
string : STRING;
i, integer : INTEGER;
                                                  {REWRITE THIS}
BEGIN
```

```
{
OUTPUT(" IN READPREPOMASTERFILE");
}
NEW(File);
ASK File TO Open(MasterFile, Input);
ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i := 1 TO integer

        NEW(prepo);
        ASK File TO ReadString(string);
        ASK prepo TO SetName(string);
        ASK PrepoQ TO Add(prepo);
        ASK File TO ReadLine(string);
END FOR;
ASK File TO Close;
DISPOSE(File);
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SavePrepoDataFile(IN Name : STRING; IN transporter : TransporterObj;
         IN shipment : ShipmentObj; IN cargo : CommodityQObj);
{---------------------------------------------------------------------------}

{BUILDS A PREPO DATAFILE}

VAR

object : NamedObj;
i, numItems : INTEGER;
file : StreamObj;
string, string2 : STRING;
base : BaseObj;
commodity : CommodityObj;

BEGIN

string := SUBSTR(1,8,Name);
string := string + ".dat" ;
NEW(file);
ASK file TO Open(string, Output);
ASK file TO WriteLn;

ASK file TO WriteString(Name);
ASK file TO WriteLn;

ASK file TO WriteLn;

ASK file TO WriteString("Transporter:  " + transporter.Name);
ASK file TO WriteLn;

ASK file TO WriteString("Latitude:  ");
ASK file TO WriteInt(transporter.Position.LatDeg,3);
ASK file TO WriteInt(transporter.Position.LatMin,3);
ASK file TO WriteString(" " + transporter.Position.LatDir);
ASK file TO WriteLn;
ASK file TO WriteString("Longitude: ");
ASK file TO WriteInt(transporter.Position.LongDeg,3);
ASK file TO WriteInt(transporter.Position.LongMin,3);
ASK file TO WriteString(" " + transporter.Position.LongDir);
ASK file TO WriteLn;

ASK file TO WriteLn;

ASK file TO WriteString("Cargo Destination:  " + shipment.Destination.Name);
ASK file TO WriteLn;

numItems := ASK shipment.Route numberIn;
ASK file TO WriteString("Cargo Routing:  " + INTTOSTR(numItems));
ASK file TO WriteLn;

FOR i := 1 TO numItems
        base := ASK shipment.Route TO Remove;
        ASK shipment.Route TO Add(base);
        ASK file TO WriteString(base.Name);
        ASK file TO WriteLn;
END FOR;
ASK file TO WriteLn;


numItems := ASK cargo numberIn;
ASK file TO WriteString("Cargo:  " + INTTOSTR(numItems));
ASK file TO WriteLn;

FOR i := 1 TO numItems
        commodity := ASK cargo TO Remove;
        ASK cargo TO Add(commodity);
        ASK file TO WriteString(commodity.Name + "  "
                 + REALTOSTR(commodity.OnHand));
        ASK file TO WriteLn;
END FOR;
ASK file TO WriteLn;

ASK file TO Close;
DISPOSE(file);

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SaveMasterTransporterFile;
{---------------------------------------------------------------------------}

{BUILDS A TRANSPORTER MASTER DATAFILE}

CONST

format = "*********>.dat";

VAR

transporter : TransporterObj;
file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
```

```
i, numItems : INTEGER;

BEGIN


NEW(file);
ASK file TO Open("Trnsprts.mst", Output);
numItems := ASK TransporterQ numberIn;
ASK file TO WriteString("NumberOfFiles: "+ INTTOSTR(numItems));
ASK file TO WriteLn;
ASK file TO WriteLn;

FOR i := 1 TO numItems
        transporter := ASK TransporterQ TO Remove;
        ASK TransporterQ TO Add(transporter);
        ASK SELF TO SaveTransporterDataFile(transporter);

        string := ASK transporter Name;
        string2 := SUBSTR(1,8,string);
        string := string2 + ".dat" ;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{-------------------------------------------------------------------------}
ASK METHOD SaveTransporterDataFile(IN Transporter : TransporterObj);
{-------------------------------------------------------------------------}


{BUILDS A TRANSPORTER DATAFILE}

VAR

file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
i, numItems : INTEGER;

BEGIN

NEW(file);

string := ASK Transporter Name;
string2 := SUBSTR(1,8,string);
string := string2 + ".dat";
ASK file TO Open(string, Output);

ASK file TO WriteLn;

ASK file TO WriteString("Type: " + Transporter.Name);
ASK file TO WriteLn;

CASE Transporter.Class
        WHEN Aircraft:
                string := "Aircraft";
```

```
          WHEN Rail:
                    string := "Rail";
          WHEN Ship:
                    string := "Ship";
          WHEN Truck:
                    string := "Truck";
END CASE;
ASK file TO WriteString("Class: " + string);
ASK file TO WriteLn;

CASE Transporter.SubClass
          WHEN Liquid:
                    string := "Liquid";
          WHEN RoRo:
                    string := "RoRo";
          WHEN BreakBulk:
                    string := "BreakBulk";
          WHEN General:
                    string := "General";
          WHEN Pax:
                    string := "Pax";
END CASE;

ASK file TO WriteString("SubClass: " + string);
ASK file TO WriteLn;

ASK file TO WriteString("Length: ");
ASK file TO WriteReal(Transporter.Length, 0, 2);
ASK file TO WriteString(" Width: ");
ASK file TO WriteReal(Transporter.Width, 0, 2);
ASK file TO WriteLn;

ASK file TO WriteString("MaxSpeed: ");
ASK file TO WriteReal(Transporter.MaxSpeed, 0, 2);
ASK file TO WriteString(" MaxRange: ");
ASK file TO WriteReal(Transporter.MaxRange, 0, 2);
ASK file TO WriteLn;

ASK file TO WriteLn;

ASK file TO WriteString("MaxCargoArea: ");
ASK file TO WriteReal(Transporter.MaxCargoArea, 0, 2);
ASK file TO WriteString(" MaxCargoCube: ");
ASK file TO WriteReal(Transporter.MaxCargoCube, 0, 2);
ASK file TO WriteString(" MaxCargoWeight: ");
ASK file TO WriteReal(Transporter.MaxCargoWeight, 0, 2);
ASK file TO WriteLn;


ASK file TO WriteString("MaxCargoLength: ");
ASK file TO WriteReal(Transporter.MaxCargoLength, 0, 2);
ASK file TO WriteString(" MaxCargoWidth: ");
ASK file TO WriteReal(Transporter.MaxCargoWidth, 0, 2);
ASK file TO WriteString(" MaxCargoHeight: ");
ASK file TO WriteReal(Transporter.MaxCargoHeight, 0, 2);
ASK file TO WriteLn;

ASK file TO WriteString(" MaxPax: ");
ASK file TO WriteReal(Transporter.MaxPax, 0, 2);
ASK file TO WriteString(" MaxGas: ");
ASK file TO WriteReal(Transporter.MaxGas, 0, 2);
ASK file TO WriteLn;

IF Transporter.OverSize
        ASK file TO WriteString("OverSize: TRUE");
ELSE
        ASK file TO WriteString("OverSize: FALSE");
END IF;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SaveMasterBaseFile;
{---------------------------------------------------------------------------}

{BUILDS A BASE MASTER DATAFILE}

CONST

format = "********>.dat";

VAR

base : BaseObj;
file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
i, numItems : INTEGER;

BEGIN

NEW(file);
ASK file TO Open("Bases.mst", Output);
numItems := ASK OnlyBaseQ numberIn;
ASK file TO WriteString("NumberOfBases: "+ INTTOSTR(numItems));
ASK file TO WriteLn;
ASK file TO WriteLn;

FOR i := 1 TO numItems
        base := ASK OnlyBaseQ TO Remove;
        ASK OnlyBaseQ TO Add(base);
        ASK SELF TO SaveBaseDataFile(base);

        string := ASK base Name;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SaveBaseDataFile(IN base : BaseObj);
{---------------------------------------------------------------------------}

{BUILDS A BASE DATAFILE}
```

```
VAR

file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
i, numItems : INTEGER;
commodity : CommodityObj;

BEGIN

NEW(file);

string := ASK base Name;
string2 := SUBSTR(1,8,string);
string := string2 + ".dat";
string := ASK base Name;
string2 := SUBSTR(1,8,string);
string := string2 + ".dat";
ASK file TO Open(string, Output);

ASK file TO WriteLn;

ASK file TO WriteString(base.Name);
ASK file TO WriteLn;

ASK file TO WriteLn;

ASK file TO WriteString("Group:  ");
CASE base.Group
WHEN CONUS:
        string := "CONUS";
WHEN ILOC:
        string := "ILOC";
WHEN THEATER:
        string := "THEATER";
OTHERWISE
        string := "THEATER";
END CASE;
ASK file TO WriteString(string);
ASK file TO WriteString(" SubGroup:  ");
CASE base.SubGroup
WHEN POE:
        string := "POE";
WHEN POS:
        string := "POS";
WHEN POD:
        string := "POD";
OTHERWISE
        string := "NONE";
END CASE;
ASK file TO WriteString(string);
ASK file TO WriteLn;

ASK file TO WriteLn;

ASK file TO WriteString("Latitude: ");
ASK file TO WriteInt(base.Position.LatDeg,3);
ASK file TO WriteInt(base.Position.LatMin,4);
ASK file TO WriteString(" " + base.Position.LatDir);
ASK file TO WriteLn;

ASK file TO WriteString("Longitude: ");
ASK file TO WriteInt(base.Position.LongDeg,3);
ASK file TO WriteInt(base.Position.LongMin,3);
ASK file TO WriteString(" " + base.Position.LongDir);
ASK file TO WriteLn;

ASK file TO WriteLn;

IF base.HasAirPort
        ASK file TO WriteString("HasAirport: TRUE ");
ELSE
        ASK file TO WriteString("HasAirport: FALSE ");
END IF;
ASK file TO WriteString(" MaxCapacity: ");
ASK file TO WriteInt(base.AirPort.MaxCapacity,0);
ASK file TO WriteString(" MaxSize: ");
ASK file TO WriteReal(base.AirPort.MaxSize, 0, 2);
ASK file TO WriteLn;

IF base.HasSeaPort
        ASK file TO WriteString("HasSeaport: TRUE ");
ELSE
        ASK file TO WriteString("HasSeaport: FALSE ");
END IF;
ASK file TO WriteString(" MaxCapacity: ");
ASK file TO WriteInt(base.SeaPort.MaxCapacity,0);
ASK file TO WriteString(" MaxSize: ");
ASK file TO WriteReal(base.SeaPort.MaxSize, 0, 2);
ASK file TO WriteLn;

IF base.HasRail
        ASK file TO WriteString("HasRail: TRUE ");
ELSE
        ASK file TO WriteString("HasRail: FALSE ");
END IF;
ASK file TO WriteString(" MaxCapacity: ");
ASK file TO WriteInt(base.RailYard.MaxCapacity,0);
ASK file TO WriteString(" MaxSize: ");
ASK file TO WriteReal(base.RailYard.MaxSize, 0, 2);
ASK file TO WriteLn;

IF base.HasTruckStop
        ASK file TO WriteString("HasTruckStop: TRUE ");
ELSE
        ASK file TO WriteString("HasTruckStop: FALSE ");
END IF;
ASK file TO WriteString(" MaxCapacity: ");
ASK file TO WriteInt(base.TruckStop.MaxCapacity,0);
ASK file TO WriteString(" MaxSize: ");
ASK file TO WriteReal(base.TruckStop.MaxSize, 0, 2);
ASK file TO WriteLn;

ASK file TO WriteLn;

integer := ASK SELF TO CountTransporterTypes(base.AirPort, file);
integer := integer + ASK SELF TO CountTransporterTypes(base.SeaPort, file);
integer := integer + ASK SELF TO CountTransporterTypes(base.RailYard, file);
integer := integer + ASK SELF TO CountTransporterTypes(base.TruckStop, file);

ASK file TO WriteString("TransporterTypes: " + INTTOSTR(integer));
ASK file TO WriteLn;

ASK file TO WriteLn;



ASK SELF TO CountTransporters(base.AirPort, file);
ASK SELF TO CountTransporters(base.SeaPort, file);
ASK SELF TO CountTransporters(base.RailYard, file);
ASK SELF TO CountTransporters(base.TruckStop, file);
ASK file TO WriteLn;

numItems := ASK base.Inventory numberIn;
ASK file TO WriteString("Commodities: " + INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems;
        commodity := ASK base.Inventory TO Remove;
        ASK base.Inventory TO Add(commodity);
        ASK file TO WriteString(commodity.Name);
        ASK file TO WriteLn;
        ASK file TO WriteString("OnHand: ");
        ASK file TO WriteReal(commodity.OnHand, 0, 2);
        ASK file TO WriteString(" StockTo ");
        ASK file TO WriteReal(commodity.StockTo, 0, 2);
        ASK file TO WriteLn;

        ASK file TO WriteString("OrderAt: ");
        ASK file TO WriteReal(commodity.OrderAt, 0, 2);
        ASK file TO WriteString(" EmerOrderAt ");
        ASK file TO WriteReal(commodity.EmerOrderAt, 0, 2);
        ASK file TO WriteLn;

        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SaveMasterUnitFile;
{---------------------------------------------------------------------------}

{BUILDS A UNIT MASTER DATAFILE}

CONST

format = "*********";

VAR

unit : UnitObj;
file : StreamObj;
string, string2 : STRING;
```

```
integer : INTEGER;
real : REAL;
i, numItems : INTEGER;

BEGIN

NEW(file);
ASK file TO Open("Units.mst", Output);
numItems := ASK UnitQ numberIn;
ASK file TO WriteString("NumberOfUnits: "+ INTTOSTR(numItems));
ASK file TO WriteLn;
ASK file TO WriteLn;

FOR i := 1 TO numItems
        unit := ASK UnitQ TO Remove;
        ASK UnitQ TO Add(unit);
        ASK SELF TO SaveUnitDataFile(unit);

        string := ASK unit Name;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SaveMasterSubUnitFile;
{-----------------------------------------------------------------------}

{BUILDS A SUBUNIT MASTER DATAFILE}

CONST

format = "********>.dat";

VAR

subunit : SubUnitObj;
file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
i, numItems, numItems2 : INTEGER;

BEGIN

NEW(file);
ASK file TO Open("SubUnits.mst", Output);
numItems := Planes.numberIn + Ships.numberIn + Tanks.numberIn;
ASK file TO WriteString("NumberOfUnits: "+ INTTOSTR(numItems));
ASK file TO WriteLn;
ASK file TO WriteLn;
numItems2 := ASK Planes numberIn;
FOR i := 1 TO numItems2
        subunit := ASK Planes TO Remove;
        ASK Planes TO Add(subunit);
        ASK subunit TO SaveSubUnitFile;
        string := ASK subunit Name;
        string2 := SUBSTR(1,8,string);
        string := string2 + ".dat" ;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;
numItems2 := ASK Ships numberIn;
FOR i := 1 TO numItems2
        subunit := ASK Ships TO Remove;
        ASK Ships TO Add(subunit);
        ASK subunit TO SaveSubUnitFile;

        string := ASK subunit Name;
        string2 := SUBSTR(1,8,string);
        string := string2 + ".dat" ;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;
numItems2 := ASK Tanks numberIn;
FOR i := 1 TO numItems2
        subunit := ASK Tanks TO Remove;
        ASK Tanks TO Add(subunit);
        ASK subunit TO SaveSubUnitFile;

        string := ASK subunit Name;
        string2 := SUBSTR(1,8,string);
        string := string2 + ".dat" ;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SaveUnitDataFile(IN unit : UnitObj);
{-----------------------------------------------------------------------------}


{BUILDS A UNIT DATAFILE}

VAR

file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
i, numItems : INTEGER;
commodity : CommodityObj;

BEGIN

NEW(file);

string := ASK unit Name;
string2 := SUBSTR(1,8,string);
string := string2 + ".dat";
string := ASK unit Name;
string2 := SUBSTR(1,8,string);
```

```
string := string2 + ".dat";
ASK file TO Open(string, Output);

ASK file TO WriteLn;

ASK file TO WriteString(unit.Name);
ASK file TO WriteLn;

CASE unit.Class
        WHEN Air :
                ASK file TO WriteString("Class:  Air");
        WHEN Sea :
                ASK file TO WriteString("Class:  Sea");
        WHEN Land :
                ASK file TO WriteString("Class:  Land");
END CASE;
ASK file TO WriteLn;

ASK file TO WriteLn;

ASK file TO WriteString("Latitude: ");
ASK file TO WriteInt(unit.Position.LatDeg,3);
ASK file TO WriteInt(unit.Position.LatMin,4);
ASK file TO WriteString(" " + unit.Position.LatDir);
ASK file TO WriteLn;

ASK file TO WriteString("Longitude: ");
ASK file TO WriteInt(unit.Position.LongDeg,3);
ASK file TO WriteInt(unit.Position.LongMin,3);
ASK file TO WriteString(" " + unit.Position.LongDir);
ASK file TO WriteLn;

ASK file TO WriteLn;


ASK file TO WriteString("DelayUntil: ");
ASK file TO WriteReal(unit.DelayUntil, 0, 2);
ASK file TO WriteLn;


IF unit.InPlace
        ASK file TO WriteString("InPlace: TRUE");
ELSE
        ASK file TO WriteString("InPlace: FALSE");
END IF;
ASK file TO WriteLn;

ASK file TO WriteString("ActiveAt: ");
ASK file TO WriteReal(unit.ActiveAt, 0, 2);
ASK file TO WriteLn;

ASK file TO WriteString("InitialCombatIntensity: ");
CASE unit.CombatIntensity
        WHEN High :
                ASK file TO WriteString("High");
        WHEN Med :
                ASK file TO WriteString("Med");
        WHEN Low :
                ASK file TO WriteString("Low"):
        OTHERWISE
```

```
                 ASK file TO WriteString("None");
   END CASE;
   ASK file TO WriteLn;

   ASK file TO WriteLn;

   IF unit.HasAirPort
           ASK file TO WriteString("HasAirport: TRUE ");
   ELSE
           ASK file TO WriteString("HasAirport: FALSE ");
   END IF;
   ASK file TO WriteString(" MaxCapacity: ");
   ASK file TO WriteInt(unit.AirPort.MaxCapacity,0);
   ASK file TO WriteString(" MaxSize: ");
   ASK file TO WriteReal(unit.AirPort.MaxSize, 0, 2);
   ASK file TO WriteLn;

   IF unit.HasSeaPort
           ASK file TO WriteString("HasSeaport: TRUE ");
   ELSE
           ASK file TO WriteString("HasSeaport: FALSE ");
   END IF;
   ASK file TO WriteString(" MaxCapacity: ");
   ASK file TO WriteInt(unit.SeaPort.MaxCapacity,0);
   ASK file TO WriteString(" MaxSize: ");
   ASK file TO WriteReal(unit.SeaPort.MaxSize, 0, 2);
   ASK file TO WriteLn;

   IF unit.HasRail
           ASK file TO WriteString("HasRail: TRUE ");
   ELSE
           ASK file TO WriteString("HasRail: FALSE ");
   END IF;
   ASK file TO WriteString(" MaxCapacity: ");
   ASK file TO WriteInt(unit.RailYard.MaxCapacity,0);
   ASK file TO WriteString(" MaxSize: ");
   ASK file TO WriteReal(unit.RailYard.MaxSize, 0, 2);
   ASK file TO WriteLn;

   IF unit.HasTruckStop
           ASK file TO WriteString("HasTruckStop: TRUE ");
   ELSE
           ASK file TO WriteString("HasTruckStop: FALSE ");
   END IF;
   ASK file TO WriteString(" MaxCapacity: ");
   ASK file TO WriteInt(unit.TruckStop.MaxCapacity,0);
   ASK file TO WriteString(" MaxSize: ");
   ASK file TO WriteReal(unit.TruckStop.MaxSize, 0, 2);
   ASK file TO WriteLn;

   ASK file TO WriteLn;

   integer := ASK SELF TO CountTransporterTypes(unit.AirPort, file);
   integer := integer + ASK SELF TO CountTransporterTypes(unit.SeaPort, file);
   integer := integer + ASK SELF TO CountTransporterTypes(unit.RailYard, file);
   integer := integer + ASK SELF TO CountTransporterTypes(unit.TruckStop, file);

   ASK file TO WriteString("TransporterTypes: " + INTTOSTR(integer));
   ASK file TO WriteLn;
```

```
ASK file TO WriteLn;

ASK SELF TO CountTransporters(unit.AirPort, file);
ASK SELF TO CountTransporters(unit.SeaPort, file);
ASK SELF TO CountTransporters(unit.RailYard, file);
ASK SELF TO CountTransporters(unit.TruckStop, file);
ASK file TO WriteLn;

numItems := ASK unit.Inventory numberIn;
ASK file TO WriteString("Commodities: " + INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems;
        commodity := ASK unit.Inventory TO Remove;
        ASK unit.Inventory TO Add(commodity);
        ASK file TO WriteString(commodity.Name);
        ASK file TO WriteLn;

        ASK file TO WriteString("HighRate: ");
        ASK file TO WriteReal(commodity.HighRate, 0, 2);
        ASK file TO WriteString(" MedRate ");
        ASK file TO WriteReal(commodity.MedRate, 0, 2);
        ASK file TO WriteString(" LowRate ");
        ASK file TO WriteReal(commodity.LowRate, 0, 2);
        ASK file TO WriteString(" NoneRate ");
        ASK file TO WriteReal(commodity.NoneRate, 0, 2);
        ASK file TO WriteLn;

        ASK file TO WriteString("OnHand: ");
        ASK file TO WriteReal(commodity.OnHand, 0, 2);
        ASK file TO WriteString(" StockTo ");
        ASK file TO WriteReal(commodity.StockTo, 0, 2);
        ASK file TO WriteLn;

        ASK file TO WriteString("OrderAt: ");
        ASK file TO WriteReal(commodity.OrderAt, 0, 2);
        ASK file TO WriteString(" EmerOrderAt ");
        ASK file TO WriteReal(commodity.EmerOrderAt, 0, 2);
        ASK file TO WriteLn;

        IF commodity.Deployment
                ASK file TO WriteString("Deployment: TRUE");
        ELSE
                ASK file TO WriteString("Deployment: FALSE");
        END IF;
        ASK file TO WriteLn;

        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD SaveMasterCommodityFile;
{----------------------------------------------------------------------------}
```

```
{BUILDS A COMMODITY MASTER DATAFILE}

VAR

commodity : CommodityObj;
file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
i, numItems : INTEGER;

BEGIN

NEW(file);
ASK file TO Open("Commods.mst", Output);
numItems := ASK CommodityQ numberIn;

ASK file TO WriteLn;
ASK file TO WriteString("NumberOfCommodities: "+ INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        commodity := ASK CommodityQ TO Remove;
        ASK CommodityQ TO Add(commodity);
        ASK file TO WriteString("Name: " + commodity.Name);
        CASE  commodity.Class
                WHEN Fuel:
                        string := "Fuel";
                WHEN FFV:
                        string := "FFV";
                WHEN Ammo:
                        string := "Ammo";
                WHEN Spares:
                        string := "Spares";
                WHEN Personnel:
                        string := "Personnel";
                WHEN Medical:
                        string := "Medical";
                WHEN Major:
                        string := "Major";
                WHEN Other:
                        string := "Other";
        END CASE;
        ASK file TO WriteString(" Class: " + string);
        ASK file TO WriteString(" ProduceAt: ");
        ASK file TO WriteReal(commodity.ProduceAt, 0, 2);
        ASK file TO WriteLn;

        ASK file TO WriteString("Length: ");
        ASK file TO WriteReal(commodity.Length, 0, 2);
        ASK file TO WriteString(" Width: ");
        ASK file TO WriteReal(commodity.Width, 0, 2);
        ASK file TO WriteString(" Height: ");
        ASK file TO WriteReal(commodity.Height, 0, 2);
        ASK file TO WriteLn;

        ASK file TO WriteString("Weight: ");
        ASK file TO WriteReal(commodity.Weight, 0, 2);
        ASK file TO WriteString(" Priority: ");
        ASK file TO WriteInt(commodity.NormalPriority, 0);
        ASK file TO WriteString(" EmerPriority: ");
        ASK file TO WriteInt(commodity.EmerPriority, 0);
        ASK file TO WriteLn;

        IF commodity.OverSize
                ASK file TO WriteString("OutSize: TRUE");
        ELSE
                ASK file TO WriteString("OutSize: FALSE");
        END IF;
        ASK file TO WriteLn;

        ASK file TO WriteLn;

END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SaveScenarioMasterFile;
{-----------------------------------------------------------------------------}

{BUILDS A SCENARIO MASTER DATAFILE}

CONST

format = "********>.dat";

VAR

scene : ScenarioObj;
file : StreamObj;
string, string2 : STRING;
i, numItems : INTEGER;

BEGIN

NEW(file);
ASK file TO Open("Scenes.mst", Output);
numItems := ASK ScenarioList numberIn;
ASK file TO WriteString("NumberOfScenarios: "+ INTTOSTR(numItems));
ASK file TO WriteLn;
ASK file TO WriteLn;

FOR i := 1 TO numItems
        scene := ASK ScenarioList TO Remove;
        ASK ScenarioList TO Add(scene);
        string := ASK scene Name;
        string2 := SUBSTR(1,8,string);
        ASK file TO WriteString(string2);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;
```

```
{----------------------------------------------------------------------}
ASK METHOD InputHasPorts(INOUT base : BaseObj);
{----------------------------------------------------------------------}

{}

VAR

answer : STRING;

BEGIN

LOOP
        OUTPUT("     Does this base or unit have an airport? (Y or N).");
        INPUT(answer);
        IF (answer = "y") OR (answer = "Y")
                ASK base SetHasAirPort("TRUE");
                EXIT;
        ELSIF (answer = "n") OR (answer = "N")
                ASK base SetHasAirPort("FALSE");
                EXIT;
        END IF;
END LOOP;

LOOP
        OUTPUT("     Does this base or unit have an seaport? (Y or N).");
        INPUT(answer);
        IF (answer = "y") OR (answer = "Y")
                ASK base SetHasSeaPort("TRUE");
                EXIT;
        ELSIF (answer = "n") OR (answer = "N")
                ASK base SetHasSeaPort("FALSE");
                EXIT;
        END IF;
END LOOP;

LOOP
        OUTPUT("     Does this base or unit have an railyard? (Y or N).");
        INPUT(answer);
        IF (answer = "y") OR (answer = "Y")
                ASK base SetHasRail("TRUE");
                EXIT;
        ELSIF (answer = "n") OR (answer = "N")
                ASK base SetHasRail("FALSE");
                EXIT;
        END IF;
END LOOP;

LOOP
        OUTPUT("     Does this base or unit have an truck stop (Y or N).");
        INPUT(answer);
        IF (answer = "y") OR (answer = "Y")
                ASK base SetHasTruckStop("TRUE");
                EXIT;
        ELSIF (answer = "n") OR (answer = "N")
                ASK base SetHasTruckStop("FALSE");
                EXIT;
        END IF;
END LOOP;
END METHOD;

{-------------------------------------------------------------------------------}
ASK METHOD InputPort() : PortObj;
{-------------------------------------------------------------------------------}
VAR

port : PortObj;
integer : INTEGER;
real : REAL;

BEGIN

NEW(port);

OUTPUT("      Input Maximum Capacity (ie max number of ships, aircraft etc..).");
INPUT(integer);
ASK port TO SetMaxCapacity(integer);
OUTPUT("       Input Size of largest single vehicle the facility can handle.");
OUTPUT("            Aircaft - Area in Square Feet");
OUTPUT("            Ships - Length in Feet");
OUTPUT("            Trains - Length in Cars");
OUTPUT("            Trucks - Number of Vehicles in a Convoy");
INPUT(real);
ASK port TO SetMaxSize(real);

RETURN(port);

END METHOD;

{-------------------------------------------------------------------------------}
ASK METHOD InputTransporters(INOUT base : BaseObj);
{-------------------------------------------------------------------------------}

{INTERACTIVELY INPUTS A TRANSPORTER TO A BASE/UNIT}

VAR

NewTransporter, transporter : TransporterObj;
string : STRING;
j,i, integer : INTEGER;
CHR : CHAR;

BEGIN
LOOP
ClearScreen;
j:=0;
SOUTPUT(" ",j);
SOUTPUT(" ",j);
        ASK TransporterQ TO Display(j);
SOUTPUT("==================================================================


        SOUTPUT("     Input Transporter Name",j);
        INPUT(string);
        transporter := ASK TransporterQ TO FindByName(string);
        IF transporter <> NILOBJ
                SOUTPUT("     How many do you wish to deploy at this base?",j);
                INPUT(integer);
                FOR i := 1 TO integer
                NewTransporter := CLONE(transporter);
                CASE NewTransporter.Class
                        WHEN Aircraft :
                            IF base.HasAirPort
                                ASK base.AirPort TO GetArrival(NewTransporter);
                            END IF;
                        WHEN Ship :
                            IF base.HasSeaPort
                                ASK base.SeaPort TO GetArrival(NewTransporter);
                            END IF;
                        WHEN Rail :
                            IF base.HasRail
                                ASK base.RailYard TO GetArrival(NewTransporter);
                            END IF;
                        WHEN Truck :
                            IF base.HasTruckStop
                                ASK base.TruckStop TO
                                                GetArrival(NewTransporter);
                            END IF;
                END CASE;
                END FOR;
        END IF;
        SOUTPUT("     Return (N).",j);
        CHR := ReadKey();
        IF (CHR = "Y") OR (CHR = "y")
                EXIT;
        END IF;
END LOOP;
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD InputCommodities(INOUT base : BaseObj);
{---------------------------------------------------------------------------}

{INTERACTIVELY INPUTS COMMODITIES TO A BASE/UNIT}

VAR

commodity, newcommodity, commodity2 : CommodityObj;
string : STRING;
j, integer : INTEGER;
real : REAL;
CHR : CHAR;

BEGIN
LOOP
        ClearScreen;
        j := 0;
        ASK CommodityQ TO Display(j);
        SOUTPUT("     Input Commodity Name",j);
        INPUT(string);
        commodity2 := ASK base.Inventory TO FindByName(string);
        IF commodity2 <> NILOBJ
                OUTPUT("     Base already stocks " + string +
                         ".  Use Modify to change base inventory");
        ELSE
                commodity := ASK CommodityQ TO FindByName(string);
                IF commodity <> NILOBJ
                        newcommodity := CLONE(commodity);
                        SOUTPUT("     How much do you want on hand?",j);
                        INPUT(real);
                        ASK newcommodity TO SetOnHand(real);
                        SOUTPUT("     How much do you the base or unit to"
                              + " stock?",j);
                        INPUT(real);
                        ASK newcommodity TO SetStockTo(real);
                        SOUTPUT("     At what level do you want the base or"
                              +" unit to Order more of the commodity?",j);
                        INPUT(real);
                        ASK newcommodity TO SetOrderAt(real);
                        ASK newcommodity TO SetEmerOrderAt(.25 *
                              newcommodity.StockTo);
                        ASK base.Inventory TO Add(newcommodity);
                END IF;
        END IF;
        SOUTPUT("     Add another commodity (Y).",j);
        CHR := ReadKey();
        IF (CHR = "n") OR (CHR = "N")
                EXIT;
        END IF;
END LOOP;
END METHOD;

{-------------------------------------------------------------------------}
ASK METHOD InputSubUnits(INOUT unit : UnitObj; IN SubUnitQ : SubUnitQObj);
{-------------------------------------------------------------------------}

{INTERACTIVELY INPUTS SUBUNITS TO A UNIT}

VAR

commodity, commodity2, newcommodity : CommodityObj;
string : STRING;
j, integer : INTEGER;
real : REAL;
CHR : CHAR;
SubUnit : SubUnitObj;
i, numItems : INTEGER;
numSubUnits : REAL;
level : REAL;
emerlevel : REAL;

BEGIN
LOOP
        ClearScreen;
        j := 0;
        SOUTPUT("============================================================

        ASK SubUnitQ TO Display(j);

SOUTPUT("====================================================================
        OUTPUT("                 SELECTING SUBUNITS");
        OUTPUT("     Add a SubUnit (Y).");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n")
                ClearScreen;
                EXIT;
        END IF;
        LOOP
```

```
                OUTPUT("     Input Name");
                INPUT(string);
                SubUnit := ASK SubUnitQ TO FindByName(string);
                IF SubUnit <> NILOBJ
                        EXIT;
                END IF;
        END LOOP;
        OUTPUT("     How many " + string + " does this unit have?");
        INPUT(numSubUnits);
        numItems := ASK SubUnit.Inventory numberIn;
        FOR i := 1 TO numItems
                commodity := ASK SubUnit.Inventory TO Remove;
                ASK SubUnit.Inventory TO Add(commodity);
                commodity2 := ASK unit.Inventory TO FindByName(commodity.Name);
                IF commodity2 = NILOBJ
                        commodity2 := ASK CommodityQ TO
                                                FindByName(commodity.Name);
                        IF commodity2 <> NILOBJ;
                                newcommodity := CLONE(commodity2);
                                ASK unit.Inventory TO Add(newcommodity);

                        ELSE
                                OUTPUT("Commodity missing from CommodityQ");
                                EXIT;
                        END IF;
                ELSE
                        newcommodity := commodity2;
                END IF;

                ASK newcommodity TO SetOnHand(newcommodity.OnHand +
                         commodity.StockTo * numSubUnits);
                ASK newcommodity TO SetStockTo(newcommodity.StockTo +
                         commodity.StockTo * numSubUnits);
                CASE newcommodity.Class
                WHEN Ammo :
                        level := .9;
                        emerlevel := .5
                WHEN Fuel :
                        level := .8;
                        emerlevel := .6
                OTHERWISE
                        level := .75;
                        emerlevel := .25
                END CASE;
                ASK newcommodity TO SetOrderAt(newcommodity.OrderAt +
                         commodity.StockTo * numSubUnits * level);
                ASK newcommodity TO SetEmerOrderAt(newcommodity.EmerOrderAt +
                         commodity.StockTo * numSubUnits * level);
                ASK newcommodity TO SetNoneRate(newcommodity.NoneRate +
                         commodity.NoneRate * numSubUnits);
                ASK newcommodity TO SetLowRate(newcommodity.LowRate +
                         commodity.LowRate * numSubUnits);
                ASK newcommodity TO SetMedRate(newcommodity.MedRate +
                         commodity.MedRate * numSubUnits);
                ASK newcommodity TO SetHighRate(newcommodity.HighRate +
                         commodity.HighRate * numSubUnits);
        END FOR;

END LOOP;
END METHOD;
```

```
{-----------------------------------------------------------------------}
 ASK METHOD CountTransporters(IN Port : PortObj; IN file : StreamObj);
{-----------------------------------------------------------------------}

{TALLYS TRANSPORTER SO A BASE DATAFILE CAN BE MADE}

VAR

queue, queue2, queue3 : TransporterQObj;
object, object2 : TransporterObj;
i, numItems, integer : INTEGER;

BEGIN

queue := CLONE(Port.BerthsQ);
queue2 := CLONE(Port.ArrivalsQ);
queue3 := CLONE(Port.ParkedQ);
WHILE queue.numberIn > 0
        object := ASK queue TO Remove;
        ASK file TO WriteString(object.Name + " ");
        integer := 1;
        numItems := ASK queue numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue TO Remove;
                IF object2.Name = object.Name
                        integer := integer + 1;
                ELSE
                        ASK queue TO Add(object2);
                END IF;
        END FOR;
        numItems := ASK queue2 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue2 TO Remove;
                IF object2.Name = object.Name
                        integer := integer + 1;
                ELSE
                        ASK queue2 TO Add(object2);
                END IF;
        END FOR;
        numItems := ASK queue3 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue3 TO Remove;
                IF object2.Name = object.Name
                        integer := integer + 1;
                ELSE
                        ASK queue3 TO Add(object2);
                END IF;
        END FOR;
        ASK file TO WriteInt(integer,0);
        ASK file TO WriteLn;
END WHILE;
WHILE queue2.numberIn > 0
        object := ASK queue2 TO Remove;
        ASK file TO WriteString(object.Name + " ");
        integer := 1;
        numItems := ASK queue2 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue2 TO Remove;
                IF object2.Name = object.Name
```

```
                        integer := integer + 1;
                ELSE
                        ASK queue2 TO Add(object2);
                END IF;
        END FOR;
        numItems := ASK queue3 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue3 TO Remove;
                IF object2.Name = object.Name
                        integer := integer + 1;
                ELSE
                        ASK queue3 TO Add(object2);
                END IF;
        END FOR;
        ASK file TO WriteInt(integer,0);
        ASK file TO WriteLn;
END WHILE;
WHILE queue3.numberIn > 0
        object := ASK queue3 TC Remove;
        ASK file TO WriteString(object.Name + " ");
        integer := 1;
        numItems := ASK queue3 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue3 TO Remove;
                IF object2.Name = object.Name
                        integer := integer + 1;
                ELSE
                        ASK queue3 TO Add(object2);
                END IF;
        END FOR;
        ASK file TO WriteInt(integer,0);
        ASK file TO WriteLn;
END WHILE;
DISPOSE(queue3);
DISPOSE(queue2);
DISPOSE(queue);

END METHOD;

{------------------------------------------------------------------------------}
 ASK METHOD CountTransporterTypes(IN Port : PortObj; IN file : StreamObj) :
                                                                        IN EGER;
{------------------------------------------------------------------------------}

{TALLYS THE NUMBER OF TRANSPORTER TYPES SO A BASE DATAFILE CAN BE MADE}

VAR

queue, queue2, queue3 : TransporterQObj;
object, object2 : TransporterObj;
i, numItems, integer : INTEGER;

BEGIN

queue := CLONE(Port.BerthsQ);
queue2 := CLONE(Port.ArrivalsQ);
queue3 := CLONE(Port.ParkedQ);
integer := 0;

WHILE queue.numberIn > 0
```

```
        object := ASK queue TO Remove;
        integer := integer + 1;
        numItems := ASK queue numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue TO Remove;
                IF object2.Name <> object.Name
                        ASK queue TO Add(object2);
                END IF;
        END FOR;
        numItems := ASK queue2 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue2 TO Remove;
                IF object2.Name <> object.Name
                        ASK queue2 TO Add(object2);
                END IF;
        END FOR;
        numItems := ASK queue3 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue3 TO Remove;
                IF object2.Name <> object.Name
                        ASK queue3 TO Add(object2);
                END IF;
        END FOR;
END WHILE;

WHILE queue2.numberIn > 0
        object := ASK queue2 TO Remove;
        integer := integer + 1;
        numItems := ASK queue2 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue2 TO Remove;
                IF object2.Name <> object.Name
                        ASK queue2 TO Add(object2)
                END IF;
        END FOR;
        numItems := ASK queue3 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue3 TO Remove;
                IF object2.Name <> object.Name
                        ASK queue3 TO Add(object2);
                END IF;
        END FOR;
END WHILE;
WHILE queue3.numberIn > 0
        object := ASK queue3 TO Remove;
        integer := integer + 1;
        numItems := ASK queue3 numberIn;
        FOR i := 1 TO numItems
                object2 := ASK queue3 TO Remove;
                IF object2.Name <> object.Name
                        ASK queue3 TO Add(object2)
                END IF;
        END FOR;
END WHILE;

DISPOSE(queue3);
DISPOSE(queue2);
DISPOSE(queue);

RETURN(integer);
```

```
END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD ReadBaseMasterFile;
{-----------------------------------------------------------------------------}

{READ BASE MASTER FILE}

CONST

MasterFile = "Bases.mst";

VAR

File : StreamObj;
base : BaseObj;
string : STRING;
i, integer : INTEGER;

BEGIN

NEW(File);
ASK File TO Open(MasterFile, Input);


ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i := 1 TO integer

        ASK File TO ReadString(string);
        string := SUBSTR(1,8,string);
        string := string + ".dat";
        base := ASK SELF TO BuildBase(string);
        ASK BaseQ TO Add(base);
        ASK OnlyBaseQ TO Add(base);
        ASK File TO ReadLine(string);
END FOR;

ASK File TO Close;
DISPOSE(File);
END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD ReadUnitMasterFile;
{-----------------------------------------------------------------------------}

{READ UNIT MASTER FILE}

CONST

MasterFile = "Units.mst";

VAR

File : St eamObj;
```

```
unit : UnitObj;
string : STRING;
i, integer : INTEGER;

BEGIN

NEW(File);
ASK File TO Open(MasterFile, Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i := 1 TO integer

        ASK File TO ReadString(string);
        string := SUBSTR(1,8,string);
        string := string + ".dat";
        unit := ASK SELF TO BuildUnit(string);
        ASK BaseQ TO Add(unit);
        ASK UnitQ TO Add(unit);
        ASK File TO ReadLine(string);
END FOR;
ASK File TO Close;
DISPOSE(File);

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD ReadSubUnitMasterFile;
{---------------------------------------------------------------------------}

{READ SUBUNIT MASTER FILE}

CONST

MasterFile = "SubUnits.mst";

VAR

File : StreamObj;
subunit : SubUnitObj;
string : STRING;
i, integer : INTEGER;

BEGIN

NEW(File);
ASK File TO Open("SubUnits.mst", Input);

ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i := 1 TO integer
```

```
        ASK File TO ReadString(string);

        NEW(subunit);
        ASK subunit TO Build(string);
        CASE subunit.Class
                WHEN Air:
                        ASK Planes TO Add(subunit);
                WHEN Sea:
                        ASK Ships TO Add(subunit);
                WHEN Land:
                        ASK Tanks TO Add(subunit);
                OTHERWISE
                        OUTPUT("Sub unit without a class");
                        DISPOSE(subunit);
        END CASE;
        ASK File TO ReadLine(string);
END FOR;
ASK File TO Close;
DISPOSE(File);

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD ReadTransporterMasterFile;
{----------------------------------------------------------------------------}

{READ TRANSPORTER MASTER FILE}

CONST

MasterFile = "Trnsprts.mst";

VAR

File : StreamObj;
transporter : TransporterObj;
string : STRING;
i, integer : INTEGER;

BEGIN

NEW(File);
ASK File TO Open(MasterFile, Input);


ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i := 1 TO integer

        ASK File TO ReadString(string);
        transporter := ASK SELF TO BuildTransporter(string);
        IF transporter <> NILOBJ
                ASK TransporterQ TO Add(transporter);
        END IF;
        ASK File TO ReadLine(string);
END FOR;
```

```
ASK File TO Close;
DISPOSE(File);
END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD ReadCommodityMasterFile;
{-----------------------------------------------------------------------------}

{READ COMMODITY MASTER FILE}

CONST

MasterFile = "Commods.mst";

VAR

BEGIN

ASK SELF TO BuildScenarioCommodities(MasterFile, CommodityQ);

END METHOD;


{-----------------------------------------------------------------------------}
ASK METHOD ReadScenarioMasterFile;
{-----------------------------------------------------------------------------}

{READ SCEANRIO MASTER FILE}

CONST

MasterFile = "Scenes.mst";

VAR

File : StreamObj;
scene : ScenarioObj;
string : STRING;
i, integer : INTEGER;

BEGIN

NEW(File);
ASK File TO Open(MasterFile, Input);


ASK File TO ReadString(string);
ASK File TO ReadInt(integer);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i := 1 TO integer

        ASK File TO ReadString(string);
        NEW(scene);
        ASK scene TO SetName(string);
        ASK ScenarioList TO Add(scene);
        ASK File TO ReadLine(string);
END FOR;

ASK File TO Close;
DISPOSE(File);
END METHOD;


{----------------------------------------------------------------------------}
ASK METHOD ObjInit; {-----------------------------------------------------------
VAR

SubUnit : SubUnitObj;

BEGIN

INHERITED ObjInit;
ASK SELF TO BuildSupply;
NEW(Planes);
NEW(Ships);
NEW(Tanks);
NEW(ScenarioList);
NEW(PrepoQ);
ASK SELF TO ReadCommodityMasterFile;
ASK SELF TO ReadTransporterMasterFile;

ASK SELF TO ReadBaseMasterFile;
ASK SELF TO ReadUnitMasterFile;
ASK SELF TO ReadSubUnitMasterFile;
ASK SELF TO ReadScenarioMasterFile;
ASK SELF TO ReadPrepoMasterFile;
END METHOD;


{----------------------------------------------------------------------------}
ASK METHOD ObjTerminate; {------------------------------------------------------
VAR

i, k, numItems, numItems2 : INTEGER;
Scene : ScenarioObj;
base : BaseObj;
prepo : NamedObj;

BEGIN

INHERITED ObjTerminate;

numItems := ASK ScenarioList numberIn;
FOR i := 1 TO numItems
        Scene := ASK ScenarioList TO Remove;
        ASK ScenarioList TO Add(Scene);
        numItems2 := ASK Scene.BaseQ numberIn;
        FOR k := 1 TO numItems2
                ASK Scene.BaseQ TO RemoveThis(ASK Scene.BaseQ First);
        END FOR;
        numItems2 := ASK Scene.UnitQ numberIn;
        FOR k := 1 TO numItems2
                ASK Scene.UnitQ TO RemoveThis(ASK Scene.UnitQ First);
        END FOR;
        numItems2 := ASK Scene.PrepoQ numberIn;
        FOR k := 1 TO numItems2
                ASK Scene.PrepoQ TO RemoveThis(ASK Scene.PrepoQ First);
        END FOR;
END FOR;
DISPOSE(ScenarioList);
DISPOSE(PrepoQ);
DISPOSE(Planes);
DISPOSE(Ships);
DISPOSE(Tanks);

END METHOD;

END OBJECT;

{
{=========================================================================}
OBJECT ObjName
{=========================================================================}

     {METHODS}

{-------------------------------------------------------------------------}
ASK METHOD
{-------------------------------------------------------------------------}
VAR

BEGIN

END METHOD;

END OBJECT;
}

END MODULE.
```

```
DEFINITION MODULE Scene;

{Scenario Object}

{Import statements}

FROM MyQueue IMPORT NamedObj,
                    MyQueueObj;
FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM Unit IMPORT UnitObj,
                 UnitQObj;

{Type Declarations}

TYPE

{=========================================================================}
ScenarioObj  = OBJECT(NamedObj);
{=========================================================================}

{FIELDS}

BaseQ : BaseQObj;
UnitQ : BaseQObj;
PrepoQ : MyQueueObj;

{METHODS}


{X} ASK METHOD PickBases;
{X} ASK METHOD PickUnits;
{} ASK METHOD PickPrepos;

{x} ASK METHOD CreateBaseFile;
{x} ASK METHOD CreateUnitFile;
{} ASK METHOD CreatePrepoFile;

{x} ASK METHOD CreateBaseLinkFile;
{x} ASK METHOD CreateRailLinkFile;
{x} ASK METHOD CreateTruckLinkFile;
{x} ASK METHOD CreateScenarioFile;
{} ASK METHOD Modify;

{x} ASK METHOD ObjInit;
OVERRIDE
{x} ASK METHOD ObjTerminate;

END OBJECT;

{=========================================================================}
ScenarioQObj = PROTO(MyQueueObj[NamedObj : #ScenarioObj])
{=========================================================================}
END PROTO;

VAR

END MODULE.
```

```
IMPLEMENTATION MODULE Scene;

{Comments}

{Import statements}

FROM MyQueue IMPORT NamedObj,
                    MyQueueObj;
FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM Unit IMPORT UnitObj,
                 UnitQObj;
FROM IOMod IMPORT StreamObj,
                  ALL FileUseType,
                  ReadKey;
FROM ScenEd IMPORT ScenarioEditor;
FROM CRTMod IMPORT ClearScreen;
FROM Supply IMPORT Supply;
{
FROM IMPORT ;
FROM IMPORT ;
FROM IMPORT ;
}

{Definitions}


{==========================================================================}
OBJECT ScenarioObj;
{==========================================================================}

     {METHODS}



{--------------------------------------------------------------------------}
ASK METHOD PickBases;
{--------------------------------------------------------------------------}

{PICKS BASES FOR SCENARIO}

VAR

j : INTEGER;
CHR : CHAR;
string : STRING;
base : BaseObj;

BEGIN

LOOP

        j := 0;
        ClearScreen;
        OUTPUT(" ");
        OUTPUT("                                    BASE SELECTION ");
        OUTPUT(" ");
        OUTPUT("     (A)dd, (S)ubtract, (L)ist Selected Bases, (R)eturn?");
        OUTPUT(" ");
        CHR := ReadKey();
        IF (CHR = "A") OR (CHR = "a");
        OUTPUT("================================================================
                ASK ScenarioEditor.OnlyBaseQ TO Display(j);
OUTPUT("========================================================================
                OUTPUT("     Input Name of base to add.");
                INPUT(string);
                base := ASK ScenarioEditor.BaseQ TO FindByName(string);
                IF base <> NILOBJ
                        IF NOT (ASK BaseQ Includes(base))
                                ASK BaseQ TO Add(base);
                        END IF;
                END IF;
        ELSIF (CHR = "S") OR (CHR = "s");
                IF BaseQ.numberIn > 0
OUTPUT("========================================================================
                ASK BaseQ TO Display(j);
OUTPUT("========================================================================

                        OUTPUT("     Input Name of base to remove.");
                        INPUT(string);
                        base := ASK BaseQ TO FindByName(string);
                        IF base <> NILOBJ
                                ASK BaseQ TO RemoveThis(base);
                        END IF;
                END IF;
        ELSIF (CHR = "L") OR (CHR = "l");
OUTPUT("========================================================================
                ASK BaseQ TO Display(j);
OUTPUT("========================================================================
                OUTPUT("     Hit any key to continue.");
                CHR := ReadKey;

        ELSIF (CHR = "R") OR (CHR = "r");
                EXIT;
        END IF;

END LOOP;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD PickUnits;
{-----------------------------------------------------------------------------}

{PICKS UNITS FOR SCENARIO}

VAR

j : INTEGER;
CHR : CHAR;
string : STRING;
unit, unit2 : BaseObj;

BEGIN

LOOP

        j := 0;
        ClearScreen;
        OUTPUT(" ");
        OUTPUT("                              UNIT SELECTION ");
        OUTPUT(" ");
        OUTPUT("     (A)dd, (S)ubtract, (L)ist Selected Units, (R)eturn?");
        CHR := ReadKey();
        IF (CHR = "A") OR (CHR = "a");
OUTPUT("================================================================
                ASK ScenarioEditor.UnitQ TO Display(j);
OUTPUT("================================================================
                OUTPUT("     Input Name of unit to add.");
                INPUT(string);
                unit := ASK ScenarioEditor.UnitQ TO FindByName(string);
                IF unit <> NILOBJ
                        IF NOT (ASK UnitQ Includes(unit))
                                ASK UnitQ TO Add(unit);
                        END IF;
                END IF;
        ELSIF (CHR = "S") OR (CHR = "s");
                IF UnitQ.numberIn > 0

OUTPUT("================================================================
                ASK UnitQ TO Display(j);
OUTPUT("================================================================
                        OUTPUT("     Input Name of unit to remove.");
                        INPUT(string);
                        unit := ASK UnitQ TO FindByName(string);
                        IF unit <> NILOBJ
                                ASK UnitQ TO RemoveThis(unit);
                        END IF;
                END IF;
        ELSIF (CHR = "L") OR (CHR = "l");
OUTPUT("================================================================
                ASK UnitQ TO Display(j);
OUTPUT("================================================================
                OUTPUT("     Hit any key to continue.");
                CHR := ReadKey;

        ELSIF (CHR = "R") OR (CHR = "r");
                EXIT;
        END IF;

END LOOP;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD PickPrepos;
{-----------------------------------------------------------------------}

{PICKS PREPOS FOR SCENARIO}

VAR

j : INTEGER;
CHR : CHAR;
string : STRING;
prepo : NamedObj;

BEGIN

LOOP
```

```
        j := 0;
        ClearScreen;
        OUTPUT(" ");
        OUTPUT("                            PREPO SELECTION ");
        OUTPUT(" ");
        OUTPUT("      (A)dd, (S)ubtract, (L)ist Selected Prepos, (R)eturn?");
        OUTPUT(" ");
        CHR := ReadKey();
        IF (CHR = "A") OR (CHR = "a");
        OUTPUT("===========================================================
                ASK ScenarioEditor.PrepoQ TO Display(j);
OUTPUT("=================================================================
                OUTPUT("     Input Name of prepo to add.");
                INPUT(string);
                prepo := ASK ScenarioEditor.PrepoQ TO FindByName(string);
                IF prepo <> NILOBJ
                        IF NOT (ASK PrepoQ Includes(prepo))
                                ASK PrepoQ TO Add(prepo);
                        END IF;
                END IF;
        ELSIF (CHR = "S") OR (CHR = "s");
                IF PrepoQ.numberIn > 0
OUTPUT("=================================================================
                ASK PrepoQ TO Display(j);
OUTPUT("=================================================================

                        OUTPUT("     Input Name of prepo to remove.");
                        INPUT(string);
                        prepo := ASK PrepoQ TO FindByName(string);
                        IF prepo <> NILOBJ
                                ASK PrepoQ TO RemoveThis(prepo);
                        END IF;
                END IF;
        ELSIF (CHR = "L") OR (CHR = "l");
OUTPUT("=================================================================
                ASK PrepoQ TO Display(j);
OUTPUT("=================================================================
                OUTPUT("     Hit any key to continue.");
                CHR := ReadKey;

        ELSIF (CHR = "R") OR (CHR = "r");
                EXIT;
        END IF;

END LOOP;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD CreateBaseFile;
{---------------------------------------------------------------------}

{WRITES SCENARIO BASE DATAFILE}

VAR

file : StreamObj;
i, numItems, numItems2 : INTEGER;
base : BaseObj;
```

```
string, string2 : STRING;
supply : BaseObj;

BEGIN
NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".bse", Output);

numItems := ASK BaseQ numberIn;
ASK file TO WriteString("NumBases:  "+ INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        base := ASK BaseQ TO Remove;
        ASK BaseQ TO Add(base);
        string := ASK base Name;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD CreateUnitFile;
{--------------------------------------------------------------------------}

{WRITES SCENARIO UNIT DATAFILE}

VAR

file : StreamObj;
i, numItems : INTEGER;
unit : BaseObj;
string, string2 : STRING;

BEGIN
NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".unt", Output);

numItems := ASK UnitQ numberIn;

ASK file TO WriteString("NumUnits:  "+ INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        unit := ASK UnitQ TO Remove;
        ASK UnitQ TO Add(unit);
        string := ASK unit Name;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
```

```
END METHOD;

{------------------------------------------------------------------------}
ASK METHOD CreatePrepoFile;
{------------------------------------------------------------------------}

{WRITES SCENARIO PREPO DATAFILE}

VAR

file : StreamObj;
i, numItems, numItems2 : INTEGER;
prepo : NamedObj;
string, string2 : STRING;
supply : BaseObj;

BEGIN
NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".ppo", Output);

numItems := ASK PrepoQ numberIn;
ASK file TO WriteString("NumPrepo:  "+ INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        prepo := ASK PrepoQ TO Remove;
        ASK PrepoQ TO Add(prepo);
        string := ASK prepo Name;
        ASK file TO WriteString(string);
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);
END METHOD;

{------------------------------------------------------------------------}
ASK METHOD CreateBaseLinkFile;
{------------------------------------------------------------------------}

{INTERACTIVELY WRITES SCENARIO UNIT LINK DATAFILE, WHILE USER PICKS UNIT
ORIGINS}

VAR

file : StreamObj;
i, j, numItems, Total : INTEGER;
currentbase, base : BaseObj;
currentunit, unit : BaseObj;
string, string2 : STRING;
supply : BaseObj;


BEGIN
NEW(file);
string := SUBSTR(1,8,Name);
```

```
ASK file TO Open(string + ".blk", Output);

numItems := ASK UnitQ numberIn;
ASK file TO WriteString("NumUnits:  " + INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        currentunit := ASK UnitQ TO Remove;
        ASK UnitQ TO Add(currentunit);

        ASK file TO WriteString("Unit:  " + currentunit.Name);
        ASK file TO WriteLn;

        ASK file TO WriteLn;

        LOOP
                OUTPUT("     Input the origin of ", currentunit.Name);
                INPUT(string);
                base := ASK BaseQ TO FindByName(string);
                IF base <> NILOBJ
                        ASK file TO WriteString("Origin:  " + base.Name);
                        EXIT;
                END IF;
        END LOOP;
        ASK file TO WriteLn;
        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD CreateRailLinkFile;
{----------------------------------------------------------------------}

{INTERACTIVELY WRITES SCENARIO RAIL LINK DATAFILE, WHILE USER BUILDS RAIL
NETWORK}

VAR

file : StreamObj;
i, j, k, numItems, numItems2, Total : INTEGER;
currentbase, base : BaseObj;
currentunit, unit : BaseObj;
string, string2 : STRING;
CHR : CHAR;


BEGIN

NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".rlk", Output);

numItems := ASK BaseQ numberIn;
Total := numItems + ASK UnitQ numberIn;
```

```
ASK file TO WriteString("NumBases:  " + INTTOSTR(Total));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        currentbase := ASK BaseQ TO Remove;
        ASK BaseQ TO Add(currentbase);

        ASK file TO WriteString("Base:  " + currentbase.Name);
        ASK file TO WriteLn;

        IF currentbase.HasRail
        LOOP
                ClearScreen;
                j := 0;
                OUTPUT(" ");
                OUTPUT("                                   BUILDING RAIL NETWORK ");


                OUTPUT(" ");
                OUTPUT("     Current Base is:  ", currentbase.Name);
                OUTPUT(" ");
                OUTPUT("     Choose bases that can be reached from this base by
                OUTPUT(" ");
                OUTPUT("     Show (B)ases, Show (N)etwork, (A)dd Base, (S)ubtrac
                CHR := ReadKey();
                IF (CHR = "B") OR (CHR = "b");
                        ASK BaseQ TO Display(j);
                        ASK UnitQ TO Display(j);
                        OUTPUT("     Hit any key to continue");
                        CHR := ReadKey();
                ELSIF (CHR = "N") OR (CHR = "n");
                        ASK currentbase.RailYard.Network TO Display(j);
                        OUTPUT("     Hit any key to continue");
                        CHR := ReadKey();
                ELSIF (CHR = "A") OR (CHR = "a");
                        LOOP
                                ClearScreen;
                                j := 0;
                                ASK BaseQ TO Display(j);
                                ASK UnitQ TO Display(j);
                                OUTPUT("     Enter base name.");
                                INPUT(string);
                                base := ASK BaseQ TO FindByName(string);
                                IF base <> NILOBJ
                                  ASK currentbase.RailYard.Network TO Add(base);
                                  EXIT;
                                ELSE
                                        base := ASK UnitQ TO FindByName(string);
                                        IF base <> NILOBJ
                                             ASK currentbase.RailYard.Network TO
                                                                  Add(base);
                                             EXIT;
                                        END IF;

                                END IF;
                        END LOOP;
                ELSIF (CHR = "S") OR (CHR = "s");
                        IF currentbase.RailYard.Network.numberIn > 0
                        LOOP
                                ClearScreen;
                                j := 0;
                                ASK currentbase.RailYard.Network TO Display(j);
                                OUTPUT("     Enter name of base to remove.");
                                INPUT(string);
                                base := ASK currentbase.RailYard.Network TO
                                                        FindByName(string);
                                IF base <> NILOBJ
                                  ASK currentbase.RailYard.Network TO
                                                        RemoveThis(base);
                                  EXIT;
                                END IF;
                        END LOOP;
                        END IF;
                ELSIF (CHR = "R") OR (CHR = "r");
                        EXIT;
                END IF;
        END LOOP;

        {eliminate all bases not in BaseQ or UnitQ}
        numItems2 := ASK currentbase.RailYard.Network numberIn;
        FOR k := 1 TO numItems2
                base := ASK currentbase.RailYard.Network TO Remove;
                IF (ASK BaseQ Includes(base)) OR (ASK UnitQ Includes(base))
                        ASK currentbase.RailYard.Network TO Add(base);
                END IF;
        END FOR;

        numItems2 := ASK currentbase.RailYard.Network numberIn;
        ASK file TO WriteString("NumNodes:  " +
                        INTTOSTR(numItems2));
        ASK file TO WriteLn;

        ASK file TO WriteLn;

        FOR k := 1 TO numItems2
                base := ASK currentbase.RailYard.Network TO Remove;
                ASK currentbase.RailYard.Network TO Add(base);
                ASK file TO WriteString(base.Name);
                ASK file TO WriteLn;
        END FOR;
        ELSE
                ASK file TO WriteString("NumNodes:  0");
                ASK file TO WriteLn;
                ASK file TO WriteLn;
        END IF;
        ASK file TO WriteLn;
END FOR;

numItems := ASK UnitQ numberIn;
FOR i := 1 TO numItems
        currentunit := ASK UnitQ TO Remove;
        ASK UnitQ TO Add(currentunit);

        ASK file TO WriteString("Base:  " + currentunit.Name);
        ASK file TO WriteLn;

        IF currentunit.HasRail
```

```
LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT("                             BUILDING RAIL NETWORK ");


        OUTPUT(" ");
        OUTPUT("     Current Base is:  ",currentunit.Name);
        OUTPUT(" ");
        OUTPUT("     Choose bases that can be reached from this base by
        OUTPUT(" ");


        ASK file TO WriteLn;

        OUTPUT("     Show (B)ases, Show (N)etwork, (A)dd Base, (S)ubtrac
        CHR := ReadKey();
        IF (CHR = "B") OR (CHR = "b");
                ASK BaseQ TO Display(j);
                OUTPUT("     Hit any key to continue");
                CHR := ReadKey();
        ELSIF (CHR = "N") OR (CHR = "n");
                ASK currentunit.RailYard.Network TO Display(j);
                OUTPUT("     Hit any key to continue");
                CHR := ReadKey();
        ELSIF (CHR = "A") OR (CHR = "a");
                LOOP
                        ClearScreen;
                        j := 0;
                                ASK BaseQ TO Display(j);
                        OUTPUT("     Enter base name.");
                        INPUT(string);
                        base := ASK BaseQ TO FindByName(string);
                        IF base <> NILOBJ
                          ASK currentunit.RailYard.Network TO Add(base);
                          EXIT;
                        END IF;
                END LOOP;
        ELSIF (CHR = "S") OR (CHR = "s");
                IF currentbase.RailYard.Network.numberIn > 0
                LOOP
                        ClearScreen;
                        j := 0;
                        ASK currentunit.RailYard.Network TO Display(j);
                        OUTPUT("     Enter name of base to remove.");
                        INPUT(string);
                        base := ASK currentunit.RailYard.Network TO
                                                FindByName(string);
                        IF base <> NILOBJ
                                ASK currentbase.RailYard.Network TO
                                                RemoveThis(base);
                        END IF;
                END LOOP;
                END IF;
        ELSIF (CHR = "R") OR (CHR = "r");
                EXIT;
        END IF;
END LOOP;
```

```
        {eliminate all bases not in BaseQ or UnitQ}
        numItems2 := ASK currentunit.RailYard.Network numberIn;
        FOR k := 1 TO numItems2
                base := ASK currentunit.RailYard.Network TO Remove;
                IF (ASK BaseQ Includes(base)) OR (ASK UnitQ Includes(base))
                        ASK currentunit.RailYard.Network TO Add(base);
                END IF;
        END FOR;
        numItems2 := ASK currentunit.RailYard.Network numberIn;
        ASK file TO WriteString("NumNodes:  " +
                        INTTOSTR(numItems2));
        ASK file TO WriteLn;

        ASK file TO WriteLn;

        FOR k := 1 TO numItems2
                base := ASK currentunit.RailYard.Network TO Remove;
                ASK currentunit.RailYard.Network TO Add(base);
                ASK file TO WriteString(base.Name);
                ASK file TO WriteLn;
        END FOR;
        ELSE
                ASK file TO WriteString("NumNodes:  0");
                ASK file TO WriteLn;
                ASK file TO WriteLn;

        END IF;

        ASK file TO WriteLn;
END FOR;

ASK file TO Close;
DISPOSE(file);

END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD CreateTruckLinkFile;
{--------------------------------------------------------------------------}

{INTERACTIVELY WRITES SCENARIO ROAD LINK DATAFILE, WHILE USER BUILDS ROAD
NETWORK}

VAR

file : StreamObj;
i, j, k, numItems, numItems2, Total : INTEGER;
currentbase, base : BaseObj;
currentunit, unit : BaseObj;
string, string2 : STRING;
CHR : CHAR;


BEGIN

NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".tlk", Output);
```

```
numItems := ASK BaseQ numberIn;
Total := numItems + ASK UnitQ numberIn;
ASK file TO WriteString("NumBases:  " + INTTOSTR(Total));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems
        currentbase := ASK BaseQ TO Remove;
        ASK BaseQ TO Add(currentbase);
        ASK file TO WriteString("Base:  " + currentbase.Name);
        ASK file TO WriteLn;

        IF currentbase.HasTruckStop
        LOOP
                ClearScreen;
                j := 0;
                OUTPUT(" ");
                OUTPUT("                            BUILDING ROAD NETWORK ");


                OUTPUT(" ");
                OUTPUT("     Current Base is:  ",currentbase.Name);
                OUTPUT(" ");
                OUTPUT("     Choose bases that can be reached from this base by
                OUTPUT(" ");
                OUTPUT("     Show (B)ases, Show (N)etwork, (A)dd Base, (S)ubtrac
                CHR := ReadKey();
                IF (CHR = "B") OR (CHR = "b");
                        ASK BaseQ TO Display(j);
                        ASK UnitQ TO Display(j);
                        OUTPUT("     Hit any key to continue");
                        CHR := ReadKey();
                ELSIF (CHR = "N") OR (CHR = "n");
                        ASK currentbase.TruckStop.Network TO Display(j);
                        OUTPUT("     Hit any key to continue");
                        CHR := ReadKey();
                ELSIF (CHR = "A") OR (CHR = "a");
                        LOOP
                                ClearScreen;
                                j := 0;
                                        ASK BaseQ TO Display(j);
                                        ASK UnitQ TO Display(j);
                                OUTPUT("     Enter base name.");
                                INPUT(string);
                                base := ASK BaseQ TO FindByName(string);
                                IF base <> NILOBJ
                                        ASK currentbase.TruckStop.Network TO
                                                                Add(base);
                                        EXIT;
                                ELSE
                                        base := ASK UnitQ TO FindByName(string);
                                        IF base <> NILOBJ
                                             ASK currentbase.TruckStop.Network
                                                                 TO Add(base);
                                             EXIT;
                                        END IF;
                                END IF;
                        END LOOP;
                ELSIF (CHR = "S") OR (CHR = "s");
```

```
                    IF currentbase.TruckStop.Network.numberIn > 0
                    LOOP
                            ClearScreen;
                            j := 0;
                            ASK currentbase.TruckStop.Network TO Display(j);
                            OUTPUT("     Enter name of base to remove.");
                            INPUT(string);
                            base := ASK currentbase.TruckStop.Network TO
                                                    FindByName(string);
                            IF base <> NILOBJ
                              ASK currentbase.TruckStop.Network TO
                                                    RemoveThis(base);
                              EXIT;
                            END IF;
                    END LOOP;
                    END IF;
            ELSIF (CHR = "R") OR (CHR = "r");
                    EXIT;
            END IF;
    END LOOP;

    numItems2 := ASK currentbase.TruckStop.Network numberIn;
    FOR k := 1 TO numItems2
            base := ASK currentbase.TruckStop.Network TO Remove;
            IF (ASK BaseQ Includes(base)) OR (ASK UnitQ Includes(base))
                    ASK currentbase.TruckStop.Network TO Add(base);
            END IF;
    END FOR;

    numItems2 := ASK currentbase.TruckStop.Network numberIn;
    ASK file TO WriteString("NumNodes:  " +
                    INTTOSTR(numItems2));
    ASK file TO WriteLn;

    ASK file TO WriteLn;

    FOR k := 1 TO numItems2
            base := ASK currentbase.TruckStop.Network TO Remove;
            ASK currentbase.TruckStop.Network TO Add(base);
            ASK file TO WriteString(base.Name);
            ASK file TO WriteLn;
    END FOR;
    ELSE
            ASK file TO WriteString("NumNodes:  0");
            ASK file TO WriteLn;
            ASK file TO WriteLn;
    END IF;
    ASK file TO WriteLn;

END FOR;

numItems := ASK UnitQ numberIn;
FOR i := 1 TO numItems
        currentunit := ASK UnitQ TO Remove;
        ASK UnitQ TO Add(currentunit);
        ASK file TO WriteString("Base:  " + currentunit.Name);
        ASK file TO WriteLn;


        IF currentunit.HasTruckStop
```

```
LOOP
        ClearScreen;
        j := 0;
        OUTPUT(" ");
        OUTPUT("                          BUILDING ROAD NETWORK ");


        OUTPUT(" ");
        OUTPUT("     Current Base is:  ",currentunit.Name);
        OUTPUT(" ");
        OUTPUT("     Choose bases that can be reached from this base by
        OUTPUT(" ");


        OUTPUT("     Show (B)ases, Show (N)etwork, (A)dd Base, (S)ubtrac
        CHR := ReadKey();
        IF (CHR = "B") OR (CHR = "b");
                ASK BaseQ TO Display(j);
                ASK UnitQ TO Display(j);
                OUTPUT("     Hit any key to continue");
                CHR := ReadKey();
        ELSIF (CHR = "N") OR (CHR = "n");
                ASK currentunit.TruckStop.Network TO Display(j);
                OUTPUT("     Hit any key to continue");
                CHR := ReadKey();
        ELSIF (CHR = "A") OR (CHR = "a");
                LOOP
                        ClearScreen;
                        j := 0;
                                ASK BaseQ TO Display(j);
                        OUTPUT("     Enter base name.");
                        INPUT(string);
                        base := ASK BaseQ TO FindByName(string);
                        IF base <> NILOBJ
                                ASK currentunit.TruckStop.Network TO
                                                        Add(base);
                                EXIT;

                        END IF;
                END LOOP;
        ELSIF (CHR = "S") OR (CHR = "s");
                IF currentunit.TruckStop.Network.numberIn > 0
                LOOP
                        ClearScreen;
                        j := 0;
                        ASK currentunit.TruckStop.Network TO Display(j);
                        OUTPUT("     Enter name of base to remove.");
                        INPUT(string);
                        base := ASK currentunit.TruckStop.Network TO
                                                FindByName(string);
                        IF base <> NILOBJ
                          ASK currentunit.TruckStop.Network TO
                                                RemoveThis(base);
                          EXIT;
                        END IF;
                END LOOP;
                END IF;
        ELSIF (CHR = "R") OR (CHR = "r");
                EXIT;
        END IF;
```

```
          END LOOP;

          {eliminate all bases not in BaseQ or UnitQ}
          numItems2 := ASK currentunit.TruckStop.Network numberIn;
          FOR k := 1 TO numItems2
                  base := ASK currentunit.TruckStop.Network TO Remove;
                  IF (ASK BaseQ Includes(base)) OR (ASK UnitQ Includes(base))
                          ASK currentunit.TruckStop.Network TO Add(base);
                  END IF;
          END FOR;
          numItems2 := ASK currentunit.TruckStop.Network numberIn;
          ASK file TO WriteString("NumNodes:  " +  INTTOSTR(numItems2));
          ASK file TO WriteLn;

          ASK file TO WriteLn;

          FOR k := 1 TO numItems2
                  base := ASK currentunit.TruckStop.Network TO Remove;
                  ASK currentunit.TruckStop.Network TO Add(base);
                  ASK file TO WriteString(base.Name);
                  ASK file TO WriteLn;
          END FOR;
          ELSE
                  ASK file TO WriteString("NumNodes:  0");
                  ASK file TO WriteLn;
                  ASK file TO WriteLn;
          END IF;
          ASK file TO WriteLn;
  END FOR;

  ASK file TO Close;
  DISPOSE(file);

  END METHOD;

  {----------------------------------------------------------------------------}
  ASK METHOD CreateScenarioFile;
  {----------------------------------------------------------------------------}

  {WRITES SCENARIO DATAFILE}

  VAR

  file : StreamObj;
  string : STRING;

  BEGIN

  NEW(file);
  string := SUBSTR(1,8,Name);
  ASK file TO Open(string + ".scn", Output);
  ASK file TO WriteString("Commods.mst");
  ASK file TO WriteLn;
  ASK file TO WriteString("Trnsprts.mst");
  ASK file TO WriteLn;
  ASK file TO WriteString(string + ".bse");
  ASK file TO WriteLn;
  ASK file TO WriteString(string + ".unt");
  ASK file TO WriteLn;
```

```
ASK file TO WriteString(string + ".blk");
ASK file TO WriteLn;

ASK file TO WriteString(string + ".rlk");
ASK file TO WriteLn;
ASK file TO WriteString(string + ".tlk");
ASK file TO WriteLn;
ASK file TO WriteString(string + ".ppo");
ASK file TO WriteLn;

ASK file TO Close;
DISPOSE(file);

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD Modify;
{-----------------------------------------------------------------------}

{ALLOWS USER TO INTERACTIVELY MODIFY EXISTING SCENARIO}

VAR

file : StreamObj;
string, string2 : STRING;
integer, i : INTEGER;
base : BaseObj;
unit : BaseObj;
prepo : NamedObj;
CHR : CHAR;

BEGIN

{Empty the ScenarioObj Base and UnitQ so that they can be refilled}

ASK BaseQ TO Empty;
ASK UnitQ TO Empty;
ASK PrepoQ TO Empty;

{Fill the BaseQ from Scenario.bse file}
NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".bse", Input);
ASK file TO ReadString(string2);
ASK file TO ReadInt(integer);
ASK file TO ReadLine(string2);

ASK file TO ReadLine(string2);

FOR i := 1 TO integer
        ASK file TO ReadString(string2);
        base := ASK ScenarioEditor.BaseQ TO FindByName(string2);
        IF base <> NILOBJ
                ASK BaseQ TO Add(base);
        END IF;
        ASK file TO ReadLine(string2);
END FOR;
DISPOSE(file);

{Fill the UnitQ from Scenario.unt file}
```

```
NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".unt", Input);
ASK file TO ReadString(string2);
ASK file TO ReadInt(integer);
ASK file TO ReadLine(string2);

ASK file TO ReadLine(string2);

FOR i := 1 TO integer
        ASK file TO ReadString(string2);
{       string2 := SUBSTR(1,(STRLEN(string2) - 4), string2);}
        unit := ASK ScenarioEditor.UnitQ TO FindByName(string2);
        ASK UnitQ TO Add(unit);
        ASK file TO ReadLine(string2);
END FOR;
DISPOSE(file);

{Fill the PrepoQ from Scenario.bse file}
NEW(file);
string := SUBSTR(1,8,Name);
ASK file TO Open(string + ".ppo", Input);
ASK file TO ReadString(string2);
ASK file TO ReadInt(integer);
ASK file TO ReadLine(string2);

ASK file TO ReadLine(string2);

FOR i := 1 TO integer
        ASK file TO ReadString(string2);
        prepo := ASK ScenarioEditor.PrepoQ TO FindByName(string2);
        IF prepo <> NILOBJ
                ASK PrepoQ TO Add(prepo);
        END IF;
        ASK file TO ReadLine(string2);
END FOR;
DISPOSE(file);

{Link the Units to their Origins}
string := SUBSTR(1,8,Name) + ".blk";
ASK ScenarioEditor TO LinkUnits(string);


{Empty the networks of all ports on the unit and base list}
integer := ASK BaseQ numberIn;
FOR i := 1 TO integer
        base := ASK BaseQ TO Remove;
        ASK BaseQ TO Add(base);
        ASK base.TruckStop.Network TO Empty;
        ASK base.RailYard.Network TO Empty;
END FOR;

integer := ASK UnitQ numberIn;
FOR i := 1 TO integer
        unit := ASK UnitQ TO Remove;
        ASK UnitQ TO Add(unit);
        ASK unit.TruckStop.Network TO Empty;
        ASK unit.RailYard.Network TO Empty;
END FOR;
```

```
{relink rail and truck networks for the scenario bases according to the scenario
string := SUBSTR(1,8,Name) + ".rlk";
ASK ScenarioEditor TO LinkRailRoads(string);
string := SUBSTR(1,8,Name) + ".tlk";
ASK ScenarioEditor TO LinkRoads(string);

{rebuild the scenario}
LOOP
ClearScreen;
OUTPUT("");
OUTPUT("     Modify?  (B)ases, (U)nits, Unit (O)rigins, (P)repos, Rai(l) Net,"
         + " Roa(d) Net, (R)eturn.");
CHR := ReadKey();

        IF (CHR = "B") OR (CHR = "b")
                ASK SELF TO PickBases;
        ELSIF (CHR = "U") OR (CHR = "u")
                ASK SELF TO PickUnits;
        ELSIF (CHR = "O") OR (CHR = "o")
                ASK SELF TO CreateBaseLinkFile;
        ELSIF (CHR = "P") OR (CHR = "p")
                ASK SELF TO PickPrepos;
        ELSIF (CHR = "L") OR (CHR = "l")
                ASK SELF TO CreateRailLinkFile;
        ELSIF (CHR = "d") OR (CHR = "D")
                ASK SELF TO CreateTruckLinkFile;
        ELSIF (CHR = "R") OR (CHR = "r")
                EXIT;
        END IF;
END LOOP;

ASK SELF TO CreateBaseFile;
ASK SELF TO CreateUnitFile;
ASK SELF TO CreatePrepoFile;
ASK SELF TO CreateScenarioFile;

END METHOD;


{------------------------------------------------------------------------}
ASK METHOD ObjInit;
{------------------------------------------------------------------------}
VAR

BEGIN

NEW(BaseQ);
NEW(UnitQ);
NEW(PrepoQ);

END METHOD;

{------------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{------------------------------------------------------------------------}
VAR

BEGIN

DISPOSE(BaseQ);
DISPOSE(UnitQ);
DISPOSE(PrepoQ);

END METHOD;

END OBJECT;

END MODULE.
```

```
DEFINITION MODULE Shpmnt;

{Shipment object is a holding object for Commodities in transit.  The Commodity
is a field of the Shipment, along with the route and destination.  RDD is
currently unused}

{Import statements}

FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM CommodQ IMPORT CommodityObj;
FROM MyQueue IMPORT NamedObj,
                    MyQueueObj;
{
FROM IMPORT
FROM IMPORT
}

{Type Declarations}

TYPE

{=========================================================================}
ShipmentObj = OBJECT(NamedObj)
{=========================================================================}

{FIELDS}

RDD : REAL;
Destination : BaseObj;
Route : BaseQObj;
Item : CommodityObj;

{METHODS}

{x} ASK METHOD SetRDD(IN NewRDD : REAL);
{x} ASK METHOD SetDestination(INOUT NewDestination : BaseObj);
{x} ASK METHOD SetRoute(INOUT NewRoute : BaseQObj);
{x} ASK METHOD SetItem(IN NewItem : CommodityObj);

{x} ASK METHOD ObjInit;
OVERRIDE
{x} ASK METHOD ObjTerminate;

END OBJECT;

{=========================================================================}
ShipmentQObj = PROTO(MyQueueObj[NamedObj : #ShipmentObj])
{=========================================================================}
OVERRIDE
{x} ASK METHOD ObjTerminate;

END PROTO;

VAR

END MODULE.
```

```
IMPLEMENTATION MODULE Shpmnt;

{Comments}

{Import statements}

FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM CommodQ IMPORT CommodityObj;
FROM MyQueue IMPORT NamedObj,
                    MyQueueObj;

{
FROM IMPORT ;
}

{Definitions}


{=============================================================================}
OBJECT ShipmentObj;
{=============================================================================}

    {METHODS}

{-----------------------------------------------------------------------------}
ASK METHOD SetRDD(IN NewRDD : REAL);
{-----------------------------------------------------------------------------}
VAR

BEGIN

RDD := NewRDD;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SetDestination(INOUT NewDestination : BaseObj);
{-----------------------------------------------------------------------------}
VAR

BEGIN

Destination := NewDestination;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SetRoute(INOUT NewRoute : BaseQObj);
{-----------------------------------------------------------------------------}
VAR

BEGIN

IF Route = NILOBJ
        DISPOSE(Route);
        Route := CLONE(NewRoute);
ELSE
        ASK Route TO Empty;
        DISPOSE(Route);
        Route := CLONE(NewRoute);
END IF;
END METHOD;

{----------------------------------------------------------------------}
ASK METHOD SetItem(IN NewItem : CommodityObj);
{----------------------------------------------------------------------}
VAR

BEGIN

Item := CLONE(NewItem);
END METHOD;

{----------------------------------------------------------------------}
ASK METHOD ObjInit;
{----------------------------------------------------------------------}
VAR

BEGIN

NEW(Route);

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{----------------------------------------------------------------------}
VAR
i, numItems : INTEGER;
BEGIN

IF Route <> NILOBJ
        ASK Route TO Empty;
        DISPOSE(Route);
END IF;
IF Item <> NILOBJ
        DISPOSE(Item);
END IF;
INHERITED ObjTerminate;
END METHOD;

END OBJECT;

{======================================================================}
PROTO ShipmentQObj;
{======================================================================}

     {METHODS}

{----------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{----------------------------------------------------------------------}
VAR

i, j, numItems : INTEGER;
object : ShipmentObj;

BEGIN
```

```
numItems := numberIn;
FOR i := 1 TO numItems

        object := ASK SELF TO Remove;
        DISPOSE(object);
END FOR;
INHERITED ObjTerminate;

END METHOD;

END PROTO;

END MODULE.
```

```
DEFINITION MODULE Stats;

{Collects statistics for the display of Unit Supply Status and Unit Deployment S

{Import statements}

FROM ListMod IMPORT QueueList;
FROM CommodQ IMPORT CommodityClassType;
FROM Base IMPORT BaseQObj,
                 BaseObj;
FROM Unit IMPORT UnitQObj;

{Type Declarations}

TYPE

StatRecType = RECORD

Name : STRING;
Class : CommodityClassType;
Total : REAL;
Level : REAL;

END RECORD;

{========================================================================}
StatListObj = OBJECT(QueueList[ANYREC : StatRecType])
{========================================================================}

ASK METHOD FindByClass(IN Class : CommodityClassType) : StatRecType;

END OBJECT;


{========================================================================}
StatObj = OBJECT;
{========================================================================}

{FIELDS}

StatList : StatListObj;

{METHODS}

ASK METHOD CollectData(IN Base : BaseObj);
ASK METHOD CollectAllData(IN BaseQ : BaseQObj);

ASK METHOD CollectDeploymentData(IN Base : BaseObj);
ASK METHOD CollectAllDeploymentData(IN BaseQ : BaseQObj);

ASK METHOD DisplayOverView;
ASK METHOD DisplayByClass(IN Class : CommodityClassType; OUT string : STRING;
                                                         OUT percent : REAL); AS
ASK METHOD ObjTerminate;

END OBJECT;

VAR
```

```
END MODULE.
{
commodity := ASK Unit.Inventory TO Remove
CASE commodity Class
        WHEN Fuel

        WHEN FFV
        WHEN Ammo
        WHEN Spares
        WHEN Personnel
        WHEN Other

}
```

```
IMPLEMENTATION MODULE Stats;

{Comments}

{Import statements}


FROM CommodQ IMPORT ALL CommodityClassType;
FROM Base IMPORT BaseQObj,
                 BaseObj;
FROM Unit IMPORT UnitObj,
                 UnitQObj;
FROM CommodQ IMPORT CommodityObj;
FROM CRTMod IMPORT ClearScreen;
FROM SimMod IMPORT SimTime;

{Definitions}

{===========================================================================}
OBJECT StatListObj;
{===========================================================================}

{---------------------------------------------------------------------------}
ASK METHOD FindByClass(IN Class : CommodityClassType) : StatRecType;
{---------------------------------------------------------------------------}

{RETURNS A STAT RECORD GIVEN A COMMODITY CLASS}

VAR

statrec : StatRecType;
i, numItems : INTEGER;

BEGIN

numItems := ASK SELF numberIn;
FOR i := 1 TO numItems
        statrec := ASK SELF TO Remove;
        ASK SELF TO Add(statrec);
        IF statrec.Class = Class
                RETURN(statrec);
        END IF;
END FOR;

END METHOD;

END OBJECT;

{===========================================================================}
OBJECT StatObj;
{===========================================================================}

     {METHODS}

{---------------------------------------------------------------------------}
ASK METHOD CollectData(IN Base : BaseObj);
{---------------------------------------------------------------------------}

{COLLECTS INVENTORY DATA BY COMMODITY CLASS FOR A GIVEN BASE}
```

```
VAR

j,numCommodities : INTEGER;

commodity : CommodityObj;
statrec : StatRecType;

BEGIN
numCommodities := ASK Base.Inventory numberIn;
FOR j := 1 TO numCommodities
        commodity := ASK Base.Inventory TO Remove;
        ASK Base.Inventory TO Add(commodity);
        statrec := ASK StatList TO FindByClass(commodity.Class);
        statrec.Total := statrec.Total + commodity.StockTo;
        statrec.Level := statrec.Level + commodity.OnHand;
END FOR;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD CollectDeploymentData(IN Base : BaseObj);
{---------------------------------------------------------------------------}

{COLLECTS DEPLOYMENT DATA BY COMMODITY CLASS FOR A GIVEN BASE}

VAR

j,numCommodities : INTEGER;

commodity : CommodityObj;
statrec : StatRecType;

BEGIN
numCommodities := ASK Base.Inventory numberIn;
FOR j := 1 TO numCommodities
        commodity := ASK Base.Inventory TO Remove;
        ASK Base.Inventory TO Add(commodity);
        IF commodity.Deployment
                statrec := ASK StatList TO FindByClass(commodity.Class);
                statrec.Total := statrec.Total + commodity.StockTo;
                statrec.Level := statrec.Level + commodity.OnHand;
        END IF;
END FOR;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD CollectAllData(IN BaseQ : BaseQObj);
{---------------------------------------------------------------------------}

{COLLECTS INVENTORY DATA BY FOR ALL BASES IN BASEQ}

VAR

i, numBases : INTEGER;

base : BaseObj;
commodity : CommodityObj;
statrec : StatRecType;
```

```
BEGIN

numBases := ASK BaseQ numberIn;
FOR i := 1 TO numBases
        base := ASK BaseQ TO Remove;
        ASK BaseQ TO Add(base);
        ASK SELF TO CollectData(base);
END FOR;

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD CollectAllDeploymentData(IN BaseQ : BaseQObj);
{----------------------------------------------------------------------}

{COLLECTS DEPLOYMENT DATA BY FOR ALL BASES IN BASEQ}

VAR

i, numBases : INTEGER;

base : BaseObj;
commodity : CommodityObj;
statrec : StatRecType;

BEGIN

numBases := ASK BaseQ numberIn;
FOR i := 1 TO numBases
        base := ASK BaseQ TO Remove;
        ASK BaseQ TO Add(base);
        ASK SELF TO CollectDeploymentData(base);
END FOR;

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD DisplayOverView;
{----------------------------------------------------------------------}

{DISPLAYS OVERVIEW TO SCREEN}

VAR

string : STRING;
percent : REAL;

BEGIN

OUTPUT(" ");
OUTPUT("     Class I - Subsistence:  ");
ASK SELF TO DisplayByClass(FFV,string,percent);
OUTPUT("     ",string, TRUNC(percent));
OUTPUT(" ");
OUTPUT("     Class III - POL:  ");
ASK SELF TO DisplayByClass(Fuel, string, percent);
OUTPUT("     ",string, TRUNC(percent));
OUTPUT(" ");
OUTPUT("     Class V - Munitions:  ");
ASK SELF TO DisplayByClass(Ammo,string,percent);
```

```
OUTPUT("      ",string, TRUNC(percent));
OUTPUT(" ");
OUTPUT("      Class VII - Major End Items:  ");
ASK SELF TO DisplayByClass(Major,string,percent);
OUTPUT("      ",string, TRUNC(percent));
OUTPUT(" ");
OUTPUT("      Class VIII - Medical Supplies:  ");
ASK SELF TO DisplayByClass(Medical,string,percent);
OUTPUT("      ",string, TRUNC(percent));
OUTPUT(" ");
OUTPUT("      Class IX - Repair Parts:  ");
ASK SELF TO DisplayByClass(Spares,string,percent);
OUTPUT("      ",string, TRUNC(percent));
OUTPUT(" ");
OUTPUT("      Personnel:  ");
ASK SELF TO DisplayByClass(Personnel,string,percent);
OUTPUT("      ",string, TRUNC(percent));
OUTPUT(" ");
OUTPUT("      Other:  ");
ASK SELF TO DisplayByClass(Other,string,percent);
OUTPUT("      ",string, TRUNC(percent));
OUTPUT("=====================================================================

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD DisplayByClass(IN Class : CommodityClassType; OUT string : STRING;
                                                        OUT percent : REAL);
{---------------------------------------------------------------------}

{DISPLAYS CLASS DATA TO SCREEN}

VAR

statrec : StatRecType;
num, i : INTEGER;

BEGIN

statrec := ASK StatList TO FindByClass(Class);
IF statrec.Total > 0.0
        percent := (statrec.Level/statrec.Total) * 100.00;
        num := TRUNC(percent/2.0);
ELSE
        percent := 100.00;
        num := 50;
END IF;
string := "";
FOR i := 1 TO num
                string := string + "X";
END FOR;
string := string + " ";
END METHOD;

{---------------------------------------------------------------------}
ASK METHOD ObjInit;
{---------------------------------------------------------------------}
VAR

i : INTEGER;
```

```
StatRec : StatRecType;

BEGIN

NEW(StatList);
NEW(StatRec);
StatRec.Class := Fuel;
ASK StatList TO Add(StatRec);
NEW(StatRec);
StatRec.Class := FFV;
ASK StatList TO Add(StatRec);
NEW(StatRec);
StatRec.Class := Ammo;
ASK StatList TO Add(StatRec);
NEW(StatRec);
StatRec.Class := Spares;
ASK StatList TO Add(StatRec);
NEW(StatRec);
StatRec.Class := Personnel;
ASK StatList TO Add(StatRec);
NEW(StatRec);
StatRec.Class := Medical;
ASK StatList TO Add(StatRec);
NEW(StatRec);
StatRec.Class := Major;
ASK StatList TO Add(StatRec);
NEW(StatRec);
StatRec.Class := Other;
ASK StatList TO Add(StatRec);
NEW(StatRec);

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{-----------------------------------------------------------------------}
VAR

i, numItems : INTEGER;
stat : StatRecType;

BEGIN

numItems := ASK StatList numberIn;
FOR i := 1 TO numItems
        stat := ASK StatList TO Remove;
        DISPOSE(stat);
END FOR;
DISPOSE(StatList);

END METHOD;

END OBJECT;

{
{=======================================================================}
OBJECT ObjName
{=======================================================================}

     {METHODS}
{-----------------------------------------------------------------------}
ASK METHOD
{-----------------------------------------------------------------------}
VAR

BEGIN

END METHOD;

END OBJECT;
}

END MODULE.
```

```
DEFINITION MODULE SubUnit;

{SubUnit Object}

{Import statements}

FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj;

FROM Unit IMPORT ALL UnitClassType;

{Type Declarations}

TYPE

{====================== =====================================================}
SubUnitObj   = OBJECT(NamedObj)
{============================================================================}

{FIELDS}

Class : UnitClassType;
Inventory : CommodityQObj;

{METHODS}

{x}ASK METHOD SetClass(IN NewClass : STRING);
{x}ASK METHOD Build(IN FileName : STRING);
{X}ASK METHOD Display;
{x}ASK METHOD Modify;
{x}ASK METHOD Create;
{x}ASK METHOD SaveSubUnitFile;
{x}ASK METHOD ObjInit;
OVERRIDE
{x}ASK METHOD ObjTerminate;

END OBJECT;

{============================================================================}
SubUnitQObj  = PROTO(MyQueueObj[NamedObj : #SubUnitObj]);
{============================================================================}
END PROTO;


END MODULE.
```

```
IMPLEMENTATION MODULE SubUnit;

{Comments}

{Import statements}

FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj;
FROM IOMod IMPORT StreamObj,
                  ALL FileUseType;
FROM Unit IMPORT ALL UnitClassType;
FROM ScenEd IMPORT ScenarioEditor;
FROM SOUTPUT IMPORT SOUTPUT;
FROM CRTMod IMPORT ClearScreen;
FROM IOMod IMPORT ReadKey;
{
FROM IMPORT ;
}

{Definitions}


{==========================================================================}
OBJECT SubUnitObj;
{==========================================================================}

     {METHODS}

{--------------------------------------------------------------------------}
ASK METHOD SetClass(IN NewClass : STRING);
{--------------------------------------------------------------------------}

BEGIN

CASE NewClass

        WHEN "Air" :
                Class := Air;
        WHEN "Sea":
                Class := Sea;
        WHEN "Land" :
                Class := Land;
        OTHERWISE
END CASE;

END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD Build(IN FileName : STRING);
{--------------------------------------------------------------------------}

{BUILDS SUBUNIT FORM DATAFILE}

VAR

File : StreamObj;
string : STRING;
integer : INTEGER;
```

```
i , numItems : INTEGER;
commodity : CommodityObj;

BEGIN

NEW(File);
ASK File TO Open(FileName, Input);

ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK SELF TO SetName(string);
ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadString(string);
ASK SELF TO SetClass(string);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

ASK File TO ReadString(string);
ASK File TO ReadInt(numItems);
ASK File TO ReadLine(string);

ASK File TO ReadLine(string);

FOR i:= 1 TO numItems

        NEW(commodity);
        ASK Inventory TO Add(commodity);

        ASK File TO ReadString(string);
        ASK commodity TO SetName(string);
        ASK File TO ReadLine(string);

        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK commodity TO SetStockTo(real);
        ASK File TO ReadString(string);
        ASK File TO ReadString(string);
        ASK commodity TO SetDeployment(string);
        ASK File TO ReadLine(string);

        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK commodity TO SetHighRate(real);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK commodity TO SetMedRate(real);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK commodity TO SetLowRate(real);
        ASK File TO ReadString(string);
        ASK File TO ReadReal(real);
        ASK commodity TO SetNoneRate(real);
        ASK File TO ReadLine(string);

        ASK File TO ReadLine(string);
```

```
END FOR;

ASK File TO Close;
DISPOSE(File);
END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD Create;
{--------------------------------------------------------------------------}

{INTERACTIVELY FILLS SUBUNIT FIELDS AND INVENTORY}

VAR

subunit : SubUnitObj;
commodity, newcommodity : CommodityObj;
string : STRING;
real : REAL;
CHR : CHAR;
j : INTEGER;
BEGIN

ClearScreen;
OUTPUT("                              CREATING A NEW SUBUNIT");
InputName;
LOOP
        OUTPUT("     What type of subunit? (A)ir, (L)and, (S)ea.");
        CHR := ReadKey();
        IF (CHR = "A") OR (CHR = "a")
                SetClass("Air");
                EXIT;
        ELSIF (CHR = "L") OR (CHR = "l");
                SetClass("Land");
                EXIT;
        ELSIF (CHR = "S") OR (CHR = "s");
                SetClass("Sea");
                EXIT;
        END IF;
END LOOP;


LOOP

Display;

        OUTPUT("     Add Commodity? (Y)");
        CHR := ReadKey();
        IF (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF;
        ClearScreen;
        j := 0;
        ASK ScenarioEditor.CommodityQ TO Display(j);
        OUTPUT("     Input Commodity Name");
        INPUT(string);
        commodity := ASK ScenarioEditor.CommodityQ TO FindByName(string);
        IF commodity <> NILOBJ
                newcommodity := CLONE(commodity);
                OUTPUT("     How much do you want the subunit to stock?");
                INPUT(real);
                ASK newcommodity TO SetStockTo(real);

                OUTPUT("     How much do you want the subunit to consume per" +
                         " day in Heavy Combat?");
                INPUT(real);
                ASK newcommodity TO SetHighRate(real);
                OUTPUT("     How much do you want the subunit to consume per" +
                         " day in Combat?");
                INPUT(real);
                ASK newcommodity TO SetMedRate(real);
                OUTPUT("     How much do you want the subunit to consume per" +
                         " day in Light Combat?");
                INPUT(real);
                ASK newcommodity TO SetLowRate(real);
                OUTPUT("     How much do you want the subunit to consume per" +
                         " day, No Combat?");
                INPUT(real);
                ASK newcommodity TO SetNoneRate(real);

                LOOP
                OUTPUT("     Do you want this commodity to count towards" +
                         " deployment? (Y or N)");
                CHR := ReadKey();
                IF (CHR = "Y") OR (CHR = "y")
                        ASK newcommodity TO SetDeployment("TRUE");
                        EXIT;
                ELSIF (CHR = "N") OR (CHR = "n")
                        ASK newcommodity TO SetDeployment("FALSE");
                        EXIT;
                END IF;
                END LOOP;
                ASK Inventory TO Add(newcommodity);


        END IF;
END LOOP;

END METHOD;
{---------------------------------------------------------------------------}
ASK METHOD Display;
{---------------------------------------------------------------------------}

{DISPLAYS SUBUNIT DATA ON SCREEN}

CONST

format =
"*************** ******.*** ******.*** ******.*** ******.*** ********* *****";

VAR

j, i, numItems : INTEGER;
commodity : CommodityObj;
string : STRING;

BEGIN

ClearScreen;
j := 0;
```

```
SOUTPUT(" ", j);
SOUTPUT("SubUnit:  "+ Name, j);
CASE Class
        WHEN Air:
                SOUTPUT("Class:     Air", j);
        WHEN Sea:
                SOUTPUT("Class:     Sea", j);
        WHEN Land:
                SOUTPUT("Class:     Land", j);
        OTHERWISE
END CASE;
SOUTPUT(" ", j);
SOUTPUT("Name                   High     Medium        Low       None  Stock To"
         + " Deploy", j);
SOUTPUT("==================================================================
numItems := ASK Inventory numberIn;
FOR i := 1 TO numItems
        commodity := ASK Inventory TO Remove;
        ASK Inventory TO Add(commodity);
        string := SPRINT(commodity.Name, commodity.HighRate,
                 commodity.MedRate, commodity.LowRate, commodity.NoneRate,
                 commodity.StockTo, commodity.Deployment) WITH format;
        SOUTPUT(string, j);
END FOR;
SOUTPUT(" ", j);

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD Modify;
{----------------------------------------------------------------------------}

{ALLOWS INTERACTIVE MODIFICATION OF SUBUNIT FIELDS}

VAR

j : INTEGER;
string : STRING;
commodity, newcommodity : CommodityObj;
CHR : CHAR;
integer : INTEGER;
real : REAL;

BEGIN

LOOP
        ClearScreen;
        j := 0;
        Display;
        OUTPUT("     (A)dd Commodity, (M)odify Commodity, (R)eturn.");
        CHR := ReadKey();
        IF (CHR = "R") OR (CHR = "r")
                EXIT;
        ELSIF (CHR = "A") OR (CHR = "a")
                ClearScreen;
                j := 0;
                ASK ScenarioEditor.CommodityQ TO Display(j);
                OUTPUT("     Input Commodity Name");
                INPUT(string);
                commodity := ASK ScenarioEditor.CommodityQ TO
                         FindByName(string);
                IF commodity <> NILOBJ
                        newcommodity := CLONE(commodity);
                        OUTPUT("      How much do you the subunit to stock?");
                        INPUT(real);
                        ASK newcommodity TO SetStockTo(real);

                        OUTPUT("      How much do you want the subunit to" +
                                 " consume per day in Heavy Combat?");
                        INPUT(real);
                        ASK newcommodity TO SetHighRate(real);
                        OUTPUT("      How much do you want the subunit to" +
                                 " consume per day in Combat?");
                        INPUT(real);
                        ASK newcommodity TO SetMedRate(real);
                        OUTPUT("      How much do you want the subunit to" +
                                 " consume per day in Light Combat?");
                        INPUT(real);
                        ASK newcommodity TO SetLowRate(real);
                        OUTPUT("      How much do you want the subunit to" +
                                 " consume per day, No Combat?");
                        INPUT(real);
                        ASK newcommodity TO SetNoneRate(real);

                        LOOP
                        OUTPUT("      Do you want this commodity to count" +
                                " towards deployment? (Y or N)");
                        CHR := ReadKey();
                        IF (CHR = "Y") OR (CHR = "y")
                                ASK newcommodity TO SetDeployment("TRUE");
                                EXIT;
                        ELSIF (CHR = "N") OR (CHR = "n")
                                ASK newcommodity TO SetDeployment("FALSE");
                                EXIT;
                        END IF;
                        END LOOP;
                        ASK Inventory TO Add(newcommodity);
                END IF;
        ELSIF (CHR = "M") OR (CHR = "m")

        OUTPUT("      Enter Commodity Name then hit <ENTER>");
        INPUT(string);
        commodity := ASK Inventory TO FindByName(string);
        OUTPUT(" ");
        IF commodity <> NILOBJ
                LOOP
                OUTPUT("      MODIFY? (S)tocking Objective, (D)eployment," +
                         " (C)onsumption, (R)eturn");
                CHR := ReadKey();
                IF (CHR = "C") OR (CHR = "c")
                        OUTPUT("      (H)igh, (M)edium, (L)ow, (N)one");
                        CHR := ReadKey();
                        IF (CHR = "H") OR (CHR = "h")
                                OUTPUT("      How much do you the subunit" +
                                         " to consume per day in Heavy Combat?");
                                INPUT(real);
                                ASK commodity TO SetHighRate(real);
                        ELSIF (CHR = "M") OR (CHR = "m")
                                OUTPUT("      How much do you want the subunit" +
                                        " to consume per day in Combat?");
                                INPUT(real);
                                ASK commodity TO SetMedRate(real);
                        ELSIF (CHR = "L") OR (CHR = "l")
                                OUTPUT("     How much do you want the subunit" +
                                        " to consume per day in Light Combat?");
                                INPUT(real);
                                ASK commodity TO SetLowRate(real);
                        ELSIF (CHR = "N") OR (CHR = "n")
                                OUTPUT("     How much do you want the subunit" +
                                        " to consume per day, No Combat?");
                                INPUT(real);
                                ASK commodity TO SetNoneRate(real);
                        ELSIF (CHR = "R") OR (CHR = "r")
                                EXIT;
                        END IF;

                ELSIF (CHR = "S") OR (CHR = "s")
                        OUTPUT("     Input new Stocking Objective.");
                        INPUT(real);
                        ASK commodity TO SetStockTo(real);
                ELSIF (CHR = "D") OR (CHR = "d")
                        LOOP
                                OUTPUT("     Do you want this commodity to" +
                                         " count towards deployment? (Y or" +
                                         " N)?");
                                CHR := ReadKey();
                                IF (CHR = "Y") OR (CHR = "y")
                                        ASK commodity TO SetDeployment("TRUE");
                                        EXIT;
                                ELSIF (CHR = "N") OR (CHR = "n")
                                        ASK commodity TO SetDeployment("FALSE");
                                        EXIT;
                                END IF;
                        END LOOP;

                ELSIF (CHR = "R") OR (CHR = "r")
                        EXIT;
                END IF;
                END LOOP;
        END IF;

        END IF;

END LOOP;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SaveSubUnitFile;
{---------------------------------------------------------------------------}

{WRITES SUBUNIT DATAFILE}

VAR

file : StreamObj;
string, string2 : STRING;
integer : INTEGER;
real : REAL;
```

```
i, numItems : INTEGER;
commodity : CommodityObj;

BEGIN

NEW(file);

string := Name;
string2 := SUBSTR(1,8,string);
string := string2 + ".dat";
ASK file TO Open(string, Output);

ASK file TO WriteLn;

ASK file TO WriteString(Name);
ASK file TO WriteLn;

CASE Class
        WHEN Air :
                ASK file TO WriteString("Class:  Air");
        WHEN Sea :
                ASK file TO WriteString("Class:  Sea");
        WHEN Land :
                ASK file TO WriteString("Class:  Land");
END CASE;
ASK file TO WriteLn;

ASK file TO WriteLn;

numItems := ASK Inventory numberIn;
ASK file TO WriteString("Commodities: " + INTTOSTR(numItems));
ASK file TO WriteLn;

ASK file TO WriteLn;

FOR i := 1 TO numItems;
        commodity := ASK Inventory TO Remove;
        ASK Inventory TO Add(commodity);
        ASK file TO WriteString(commodity.Name);
        ASK file TO WriteLn;

        ASK file TO WriteString("StockTo ");
        ASK file TO WriteReal(commodity.StockTo, 0, 2);
        IF commodity.Deployment
                ASK file TO WriteString(" Deployment: TRUE");
        ELSE
                ASK file TO WriteString(" Deployment: FALSE");
        END IF;
        ASK file TO WriteLn;

        ASK file TO WriteString("HighRate: ");
        ASK file TO WriteReal(commodity.HighRate, 0, 3);
        ASK file TO WriteString(" MedRate ");
        ASK file TO WriteReal(commodity.MedRate, 0, 3);
        ASK file TO WriteString(" LowRate ");
        ASK file TO WriteReal(commodity.LowRate, 0, 3);
        ASK file TO WriteString(" NoneRate ");
        ASK file TO WriteReal(commodity.NoneRate, 0, 3);
        ASK file TO WriteLn;
        ASK file TO WriteLn;
    END FOR;

    ASK file TO Close;
    DISPOSE(file);
    END METHOD;

{------------------------------------------------------------------------------}
ASK METHOD ObjInit;
{------------------------------------------------------------------------------}
VAR

BEGIN

NEW(Inventory);

END METHOD;

{------------------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{------------------------------------------------------------------------------}
VAR

BEGIN

DISPOSE(Inventory);

END METHOD;

END OBJECT;


{
{==============================================================================}
OBJECT ObjName
{==============================================================================}

    {METHODS}

{------------------------------------------------------------------------------}
ASK METHOD
{------------------------------------------------------------------------------}
VAR

BEGIN

END METHOD;

END OBJECT;
}

END MODULE.
```

```
DEFINITION MODULE Supply;

{Supply Node Object}

{Import statements}

FROM Base IMPORT BaseObj;
FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;
{Type Declarations}

TYPE

{===========================================================================}
SupplyObj = OBJECT(BaseObj)
{===========================================================================}

     {Fields}

     {Methods}

        TELL METHOD Produce;
        ASK METHOD Send(INOUT Shipment : ShipmentObj; IN Qty : REAL);
        ASK METHOD SetInventory(IN NewInventory : CommodityQObj);
OVERRIDE
        ASK METHOD OrderStuff(INOUT Item : CommodityObj);
        ASK METHOD FillOrder(INOUT Shipment : ShipmentObj);
        ASK METHOD DumpFields;
        ASK METHOD ObjInit;
        ASK METHOD ObjTerminate;
        ASK METHOD ReceiveStuff(INOUT Cargo : ShipmentQObj);
        TELL METHOD CheckInventory;

END OBJECT;

VAR

Supply : SupplyObj;

END MODULE.
```

```
IMPLEMENTATION MODULE Supply;

{Supply Node Object}

{Import statements}

FROM Base IMPORT BaseObj, BaseQObj;
FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj;
FROM SimManager IMPORT StopTime;
FROM SimMod IMPORT SimTime;
FROM WriteLine IMPORT WriteLine;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;
{Definitions}

{===========================================================================}
OBJECT SupplyObj;
{===========================================================================}

{---------------------------------------------------------------------------}
     TELL METHOD Produce;
{---------------------------------------------------------------------------}

{PRODUCES ALL COMMODITIES IN THE SCENARIO EVERY 24.0 TIME UNITS}

          VAR
               CurrentStuff : CommodityObj;
               StuffName : STRING;
               DailyProduct : REAL;
          BEGIN
               LOOP
                    WAIT DURATION 24.0
                    CurrentStuff := ASK Inventory First;
                    WHILE CurrentStuff <> NILOBJ
                         StuffName := ASK CurrentStuff Name;
                         ASK CurrentStuff TO AddOnHand(CurrentStuff.ProduceAt);
                         CurrentStuff := ASK Inventory Next(CurrentStuff);
                    END WHILE;
                    END WAIT;
{WriteLine("Supply Producing Comodities");}
                ASK SELF TO FillBackOrders;
               END LOOP;
          END METHOD;

{---------------------------------------------------------------------------}
    ASK METHOD Send(INOUT Shipment : ShipmentObj; IN Qty : REAL);
{---------------------------------------------------------------------------}

{SENDS SHIPMENT OF COMMODITY DIRECTLY TO A BASE--LIKE MAGIC}

VAR

Item : CommodityObj;

     BEGIN
          Item := ASK Shipment.Destination.Inventory TO
                                                 FindByName(Shipment.Item.Name);
        IF Item <> NILOBJ
          ASK Item TO AddOnHand(Qty);
```

```
            ASK Item TO SubtractOnOrder(Qty);
          END IF;
          DISPOSE(Shipment);
       END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD SetInventory(IN NewInventory : CommodityQObj);
{---------------------------------------------------------------------------}
VAR

i, numItems : INTEGER;
newcommodity,commodity : CommodityObj;

BEGIN

numItems := ASK NewInventory numberIn;
FOR i := 1 TO numItems
        commodity := ASK NewInventory TO Remove;
        ASK NewInventory TO Add(commodity);
        newcommodity := CLONE(commodity);
        ASK Inventory TO Add(newcommodity);
END FOR;

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD DumpFields;
{---------------------------------------------------------------------------}
VAR

i, numItems : INTEGER;
Commodity : CommodityObj;
BEGIN
WriteLine(" ");
WriteLine("========================="+REALTOSTR(SimTime)+"====================
WriteLine(" ");

WriteLine("Base Name = "+ Name);
WriteLine(" ");


numItems := ASK Inventory numberIn;

WriteLine("Inventory:  "+ INTTOSTR(numItems));

FOR i := 1 TO numItems
        Commodity := ASK Inventory TO Remove;
        WriteLine(INTTOSTR(i)+". "+ Commodity.Name+": OnHand - "
      +REALTOSTR(Commodity.OnHand)+" OnOrder - "+REALTOSTR(Commodity.OnOrder)+"
        ASK Inventory TO Add(Commodity);
END FOR;

WriteLine("==============================================================");
WriteLine(" ");

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD OrderStuff(INOUT Item : CommodityObj);
{---------------------------------------------------------------------------}
```

```
     BEGIN
     END METHOD;

{------------------------------------------------------------------------------}
     ASK METHOD ReceiveStuff(INOUT Cargo : ShipmentQObj);
{------------------------------------------------------------------------------}
     BEGIN
     END METHOD;


{------------------------------------------------------------------------------}
     ASK METHOD FillOrder(INOUT Shipment : ShipmentObj);
{------------------------------------------------------------------------------}

{FILLS ORDER FROM SENT BY LOGISITCS MANAGER}


     VAR
MyItem : CommodityObj;
MyOnHand : REAL;
Receiver : BaseObj;
ReceiverItem : CommodityObj;
ReceiverItemName : STRING;
ReceiverOnHand : REAL;
ReceiverOnOrder : REAL;
ReceiverStockTo : REAL;
OrderQty : REAL;
Difference : REAL;
NewShipment : ShipmentObj;
Destination : BaseObj;
Route : BaseQObj;

     BEGIN

{determine OrderQty}

Receiver := ASK Shipment Destination;
ReceiverItem := ASK Shipment Item;
ReceiverOnHand := ASK ReceiverItem OnHand;
ReceiverOnOrder := ASK ReceiverItem OnOrder;
ReceiverStockTo := ASK ReceiverItem StockTo;
OrderQty := ReceiverStockTo - ReceiverOnHand;

IF OrderQty >= 1.0

{find item in inventory}

        ReceiverItemName := ASK ReceiverItem Name;
        MyItem :=  ASK Inventory TO FindByName(ReceiverItemName);

        IF MyItem <> NILOBJ

        MyOnHand := ASK MyItem OnHand;

{if there is sufficient On hand}

        IF MyOnHand >= OrderQty
{WriteLine("Supply sends " + REALTOSTR(OrderQty) + " " + ReceiverItem.Name + " t

                ASK SELF TO Send(Shipment, OrderQty);
```

```
                  ASK MyItem TO SubtractOnHand(OrderQty);

{Otherwise, send what there is and back order the remainder}

            ELSE
                 Difference := OrderQty - MyOnHand;

{WriteLine("Supply sends " + REALTOSTR(MyOnHand) + " " +ReceiverItem.Name +" to
                  NEW(NewShipment);
                  Destination := ASK Shipment Destination;
                  ASK NewShipment TO SetDestination(Destination);
                  ASK NewShipment TO SetRDD(Shipment.RDD);
                  Route := ASK Shipment Route;
                  ASK NewShipment TO SetRoute(Route);
                  ASK NewShipment TO SetItem(Shipment.Item);
                  ASK NewShipment.Item TO SetStockTo(Difference);
                  ASK NewShipment.Item TO SetOnOrder(0.0);
                  ASK NewShipment.Item TO SetOnHand(0.0);
                  ASK SELF TO Send(Shipment, MyOnHand);

                  ASK MyItem TO SubtractOnHand(MyOnHand);
                  ASK SELF TO BackOrderStuff(NewShipment);

            END IF;

          END IF;

ELSE
          DISPOSE(Shipment);
END IF;
     END METHOD;

{--------------------------------------------------------------------------}
     TELL METHOD CheckInventory;
{--------------------------------------------------------------------------}
     BEGIN
     END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD ObjInit;
{--------------------------------------------------------------------------}
     BEGIN

INHERITED ObjInit;
NEW(BackOrders);
SetName("Supply");
SetGroup("CONUS");
SetSubGroup("NONE");
{TELL SELF TO Produce;}

     END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD ObjTerminate;
{--------------------------------------------------------------------------}
BEGIN

INHERITED ObjTerminate;

END METHOD;
END OBJECT;

END MODULE.
```

```
DEFINITION MODULE TManage;

{Transportation Manager Object}

{Import statements}

FROM Base IMPORT BaseObj;
FROM Port IMPORT CargoGroupObj;
FROM Trnsprt IMPORT ALL TransporterClassType,
                    ALL TransporterSubClassType,
                    TransporterObj,
                    TransporterQObj;
FROM MyQueue IMPORT MyQueueObj;
FROM Distant IMPORT PositionRecType,
                    CalcDistance;
FROM GrpMod IMPORT QueueObj;
{FROM IMPORT
FROM IMPORT}

{Type Declarations}

TYPE
{=============================================================================}
RequestObj = OBJECT
{=============================================================================}

     {fields}
Requester : BaseObj;
TransporterClass : TransporterClassType;
TransporterSubClass : TransporterSubClassType;
OverSize : BOOLEAN;

     {methods}
ASK METHOD GetRequester(INOUT NewRequester : BaseObj);
ASK METHOD GetTransporterClass(IN NewTransportClass : TransporterClassType);
ASK METHOD GetTransporterSubClass(IN NewTransporterSubClass :
         TransporterSubClassType);
ASK METHOD SetOverSize(IN NewOverSize : BOOLEAN);

END OBJECT;

{=============================================================================}
RequestQObj = OBJECT(QueueObj[ANYOBJ : RequestObj]);
{=============================================================================}

OVERRIDE
ASK METHOD ObjTerminate;

END OBJECT;

{=============================================================================}
TransporterManagerObj = OBJECT
{=============================================================================}

     {FIELDS}

AvailableShips : TransporterQObj;
AvailableAircraft : TransporterQObj;
AvailableTrains : TransporterQObj;
AvailableTrucks : TransporterQObj;
```

```
AircraftRequestList : RequestQObj;
ShipRequestList : RequestQObj;
TrainRequestList : RequestQObj;
TruckRequestList : RequestQObj;

     {METHODS}

{ASK METHOD SetAvailableTransporters(IN NewTransporters : TransporterQObj);}
ASK METHOD ReceiveRequest(INOUT Asker : BaseObj; IN Class :
         TransporterClassType; IN SubClass : TransporterSubClassType; IN
         OverSize : BOOLEAN);

ASK METHOD ReceiveAvailableTransporter(INOUT Transporter : TransporterObj);
ASK METHOD CheckAvailableTransporters(INOUT AvailableTransporters :
         TransporterQObj; INOUT RequestList : RequestQObj);
ASK METHOD ObjInit;
ASK METHOD ObjTerminate;

END OBJECT;


VAR

TransporterManager : TransporterManagerObj;

END MODULE.
```

```
IMPLEMENTATION MODULE TManage;

{Transporter Manager Object}

{Import statements}

FROM Base IMPORT BaseObj;
FROM Port IMPORT CargoGroupObj;
FROM Trnsprt IMPORT ALL TransporterClassType,
                    ALL TransporterSubClassType,
                    ALL MovementStatusType,
                    TransporterObj,
                    TransporterQObj;
FROM MyQueue IMPORT MyQueueObj;
FROM Distant IMPORT PositionRecType,
                    CalcDistance;

{Definitions}

{==========================================================================}
OBJECT RequestObj;
{==========================================================================}

{METHODS}
{--------------------------------------------------------------------------}
ASK METHOD GetRequester(INOUT NewRequester : BaseObj);
{--------------------------------------------------------------------------}

     BEGIN
          Requester := NewRequester;
     END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD GetTransporterClass(IN NewTransporterClass : TransporterClassType);
{--------------------------------------------------------------------------}

     BEGIN
          TransporterClass := NewTransporterClass;
     END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD GetTransporterSubClass(IN NewTransporterSubClass :
         TransporterSubClassType);
{--------------------------------------------------------------------------}

     BEGIN
          TransporterSubClass := NewTransporterSubClass;
     END METHOD;

{--------------------------------------------------------------------------}
ASK METHOD SetOverSize(IN NewOverSize : BOOLEAN);
{--------------------------------------------------------------------------}
VAR

BEGIN

OverSize := NewOverSize;

END METHOD;
END OBJECT;

{=======================================================================}
OBJECT RequestQObj;
{=======================================================================}

{-----------------------------------------------------------------------}
ASK METHOD ObjTerminate;
{-----------------------------------------------------------------------}
VAR
i, NumItems : INTEGER;
object : ANYOBJ;

BEGIN

NumItems := numberIn;
FOR i := 1 TO numberIn
        object := Remove;
        DISPOSE(object);
END FOR;

INHERITED ObjTerminate;

END METHOD;


END OBJECT;


{=======================================================================}
OBJECT TransporterManagerObj;
{=======================================================================}

{METHODS}

{
{-----------------------------------------------------------------------}
     ASK METHOD SetAvailableTransporters(IN NewTransporters : TransporterQObj);
{-----------------------------------------------------------------------}
VAR

BEGIN
        AvailableTransporters := CLONE(NewTransporters);
END METHOD;
}

{-----------------------------------------------------------------------}
ASK METHOD ReceiveRequest(INOUT Asker : BaseObj; IN Class :
         TransporterClassType; IN SubClass : TransporterSubClassType; IN
         OverSize : BOOLEAN);
{-----------------------------------------------------------------------}

{HANDLES REQUEST FROM PORT OBJECT}

VAR

CurrentRequest : RequestObj;
RequestList : RequestQObj;
AvailableTransporters : TransporterQObj;
```

```
BEGIN

{
OUTPUT("IN ReceiveRequest - ",Asker.Name," ",Class," ", OverSize);
}
NEW(CurrentRequest);
ASK CurrentRequest TO GetRequester(Asker);
ASK CurrentRequest TO GetTransporterClass(Class);
ASK CurrentRequest TO GetTransporterSubClass(SubClass);
ASK CurrentRequest TO SetOverSize(OverSize);

CASE CurrentRequest.TransporterClass
WHEN Aircraft:
        RequestList := AircraftRequestList;
        AvailableTransporters := AvailableAircraft;
WHEN Rail:
        RequestList := TrainRequestList;
        AvailableTransporters := AvailableTrains;
WHEN Truck:
        RequestList := TruckRequestList;
        AvailableTransporters := AvailableTrucks;
WHEN Ship:
        RequestList := ShipRequestList;
        AvailableTransporters := AvailableShips;
OTHERWISE
        OUTPUT("PASSED INVALID TRANSPORTER CLASS - TRANSPORTER MANAGER");
        HALT;
END CASE;


ASK RequestList TO Add(CurrentRequest);
IF AvailableTransporters.numberIn > 0
        ASK SELF TO CheckAvailableTransporters(AvailableTransporters,
                  RequestList);
END IF;

END METHOD;

{----------------------------------------------------------------------------}
     ASK METHOD ReceiveAvailableTransporter
                         (INOUT NewTransporter : TransporterObj);
{----------------------------------------------------------------------------}

{PLACES RECEIVED TRANSPORTER IN PROPER AVAILABLE TRANSPORTER QUEUE AND CHECKS IF
IT CAN FILL ANY REQUESTS}

VAR

RequestList : RequestQObj;
AvailableTransporters : TransporterQObj;

BEGIN

{
OUTPUT("IN ReceiveAvailableTransporter - ",NewTransporter.Name,
         NewTransporter.VehicleID);
}
CASE NewTransporter.Class
WHEN Aircraft:
        RequestList := AircraftRequestList;
        AvailableTransporters := AvailableAircraft;
WHEN Rail:
        RequestList := TrainRequestList;
        AvailableTransporters := AvailableTrains;
WHEN Truck:
        RequestList := TruckRequestList;
        AvailableTransporters := AvailableTrucks;
WHEN Ship:
        RequestList := ShipRequestList;
        AvailableTransporters := AvailableShips;
OTHERWISE
        OUTPUT("PASSED INVALID TRANSPORTER CLASS - TRANSPORTER MANAGER");
        HALT;
END CASE;

ASK AvailableTransporters TO Add(NewTransporter);
ASK NewTransporter TO SetStatus(AVAILABLE);
IF RequestList.numberIn > 0
        ASK SELF TO CheckAvailableTransporters(AvailableTransporters,
                  RequestList);
END IF;

     END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD CheckAvailableTransporters(INOUT AvailableTransporters :
         TransporterQObj; INOUT RequestList : RequestQObj);
{-----------------------------------------------------------------------------}

{CHECKS TO SEE IF ANY TRANSPORTERS MEET THE NEEDS OF ANY REQUESTS}

VAR

CurrentRequest : RequestObj;
CurrentTransporter : TransporterObj;
CurrentTransporterLoc : PositionRecType;
Distance : REAL;
BestDistance : REAL;
BestTransporter : TransporterObj;
m, numRequests : INTEGER;
i, numItems : INTEGER;

BEGIN
{
OUTPUT("IN CheckAvailableTransporters");
}
{Go Thru Requests and fill each one in order}

numRequests := ASK RequestList numberIn;
FOR m := 1 TO numRequests
        CurrentRequest := ASK RequestList Remove;
        ASK RequestList TO Add(CurrentRequest);

        BestTransporter := NILOBJ;
        BestDistance := 9999999.00;

        numItems := ASK AvailableTransporters numberIn;
        FOR i := 1 TO numItems

                {Get a transporter from the AvailableTransporters}
                CurrentTransporter := ASK AvailableTransporters TO Remove;
                ASK AvailableTransporters TO Add(CurrentTransporter);

                {check if the CurrentTransporter is the right class, subclass,
                 and oversize}
                IF CurrentRequest.OverSize
                        IF CurrentTransporter.OverSize

                                {check Current Transporter distance}
                                CurrentTransporterLoc := ASK CurrentTransporter
                                         Position;
                                Distance := CalcDistance(CurrentTransporterLoc,
                                ASK CurrentRequest Requester.Position);

                                {if the distance is better than the best so far
                                 make it the best and make the transporter the
                                 best choice}
                                IF (Distance < BestDistance)
                                        BestDistance := Distance;
                                        BestTransporter := CurrentTransporter;
                                END IF;
                        END IF;
                ELSE
                        IF CurrentRequest.TransporterSubClass =
                         CurrentTransporter.SubClass
                                {check Current Transporter distance}
                                CurrentTransporterLoc := ASK
                                        CurrentTransporter Position;
                                Distance := CalcDistance(CurrentTransporterLoc,
                                         ASK CurrentRequest Requester.Position);

                                {if the distance is better than the best so far
                                 make it the best and make the transporter the
                                 best choice}
                                IF (Distance < BestDistance)
                                        BestDistance := Distance;
                                        BestTransporter := CurrentTransporter;
                                END IF;
                        END IF;
                END IF;
        END FOR;

        IF BestTransporter <> NILOBJ
                TELL BestTransporter TO GoTo(CurrentRequest.Requester);
                ASK AvailableTransporters TO RemoveThis(BestTransporter);
                ASK RequestList TO RemoveThis(CurrentRequest);
                DISPOSE(CurrentRequest);
        END IF;
END FOR;

END METHOD;

{----------------------------------------------------------------------}
     ASK METHOD ObjInit;
{----------------------------------------------------------------------}
VAR

BEGIN

NEW(AvailableShips);
NEW(ShipRequestList);

NEW(AvailableAircraft);
NEW(AircraftRequestList);

NEW(AvailableTrains);
NEW(TrainRequestList);

NEW(AvailableTrucks);
NEW(TruckRequestList);

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD ObjTerminate;
{--------------------------------------------------------------------------}
VAR

BEGIN

DISPOSE(AvailableShips);
DISPOSE(ShipRequestList);

DISPOSE(AvailableAircraft);
DISPOSE(AircraftRequestList);

DISPOSE(AvailableTrains);
DISPOSE(TrainRequestList);

DISPOSE(AvailableTrucks);
DISPOSE(TruckRequestList);

END METHOD;


END OBJECT;

END MODULE.
```

```
DEFINITION MODULE Trash;

{TrashCan and Garbage Disposal are holding queue into which discard objects are
thrown before disposal.  The TrashCan is particularly useful for DISPOSING
objects that have several TELL METHODS in progress.}

{Import statements}

FROM GrpMod IMPORT QueueObj;

{FROM IMPORT}
{FROM IMPORT}

{Type Declarations}

TYPE
{=========================================================================}
TrashCanObj  = OBJECT(QueueObj);
{=========================================================================}

{FIELDS}

{METHODS}

TELL METHOD TakeOutTheTrash;
ASK METHOD ObjInit;

END OBJECT;

{=========================================================================}
GarbageDisposalObj  = OBJECT(QueueObj);
{=========================================================================}

{FIELDS}

{METHODS}

ASK METHOD TakeOutTheGarbage;

END OBJECT;


VAR

TrashCan : TrashCanObj;
GarbageDisposal : GarbageDisposalObj;

END MODULE.
```

```
IMPLEMENTATION MODULE Trash;

{Comments}

{Import statements}
{
FROM IMPORT ;
FROM IMPORT ;
}
{Definitions}

{===================================================================}
OBJECT TrashCanObj;
{===================================================================}

     {METHODS}
{-------------------------------------------------------------------}
TELL METHOD TakeOutTheTrash;
{-------------------------------------------------------------------}

{PERIODICALLY DISPOSES OF ALL OBJECTS IN QUEUE.  TRASHCAN WAITS 100 TIME UNITS
TO ALLOW ANY TELL METHODS TO INTERRUPT}

VAR

object : ANYOBJ;
numItems, i : INTEGER;

BEGIN

WAIT DURATION 100.00

        numItems := numberIn;
        FOR i := 1 TO numItems
                object := ASK SELF TO Remove;
                DISPOSE(object);
        END FOR;

ON INTERRUPT

        numItems := numberIn;
        FOR i := 1 TO numItems
                object := ASK SELF TO Remove;
                DISPOSE(object);
        END FOR;

END WAIT

END METHOD;

{-------------------------------------------------------------------}
ASK METHOD ObjInit;
{-------------------------------------------------------------------}
VAR

BEGIN

TELL SELF TO TakeOutTheTrash;

END METHOD;
END OBJECT;

{========================================================================}
OBJECT GarbageDisposalObj;
{========================================================================}

     {METHODS}
{------------------------------------------------------------------------}
ASK METHOD TakeOutTheGarbage;
{------------------------------------------------------------------------}

{DISPOSES OF ALL OBJECTS IN THE GARBAGE DISPOSAL.  GARBAGE DISPOSAL IS MEANT TO

VAR

object : ANYOBJ;
j, numItems, i : INTEGER;

BEGIN
        {numItems := numberIn;}
        WHILE numberIn > 0
        {FOR i := 1 TO numItems}
                object := ASK SELF TO Remove;
                WHILE ASK SELF Includes(object)
                        ASK SELF TO RemoveThis(object);
                END WHILE;
                DISPOSE(object);
        {END FOR;}
        END WHILE;

END METHOD;

END OBJECT;


END MODULE.
```

```
DEFINITION MODULE Trnsprt;

{Basic Transporter Object}


{Import statements}

FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj;
FROM Node IMPORT NodeObj;
FROM Distant IMPORT PositionRecType;
FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM Base IMPORT BaseObj;
FROM Port IMPORT PortObj;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;
{Type Declarations}

CONST

PalletHeight = 96.00; {inches}

TYPE

TransporterClassType = (Aircraft, Rail, Truck, Ship);
TransporterSubClassType = (Liquid, RoRo, BreakBulk, General, Pax);
MovementStatusType = (AVAILABLE, LOADING, ENROUTE, UNLOADING);

{==========================================================================}
LoadQObj = OBJECT(ShipmentQObj);
{==========================================================================}
        {Fields}

TotalCube : REAL;
TotalArea : REAL;
TotalWeight : REAL;
TotalPax : REAL;
TotalGas : REAL;
TotalOutSize : REAL;

ASK METHOD SetTotalCube(IN NewTotalCube : REAL);
ASK METHOD SetTotalArea(IN NewTotalArea : REAL);
ASK METHOD SetTotalWeight(IN NewTotalWeight : REAL);
ASK METHOD SetTotalPax(IN NewTotalPax : REAL);
ASK METHOD SetTotalGas(IN NewTotalGas : REAL);
ASK METHOD SetTotalOutSize(IN NewTotalOutSize : REAL);

OVERRIDE
ASK METHOD Add(IN NewMember : ShipmentObj);
ASK METHOD Remove : ShipmentObj;
ASK METHOD RemoveThis(IN member : ShipmentObj);
END OBJECT;

{==========================================================================}
TransporterObj = OBJECT(NamedObj);
{==========================================================================}

     {Fields}
```

```
        {Vehicle ID Data}

        VehicleID : INTEGER;
        Class : TransporterClassType;
        SubClass : TransporterSubClassType;
        OverSize : BOOLEAN;

        {Vehicle Performance Data}

        Length : REAL;
        Width : REAL;
        MaxSpeed : REAL;
        MaxRange : REAL;
        MaxCargoCube : REAL;
        MaxCargoArea : REAL;
        MaxCargoWeight : REAL;
        MaxCargoLength : REAL;
        MaxCargoWidth : REAL;
        MaxCargoHeight : REAL;
        MaxPax : REAL;
        MaxGas : REAL;

        {Movement Data}

        Location : BaseObj;
        Position : PositionRecType;
        Destination : BaseObj;
        Status : MovementStatusType;
        Port : PortObj;
        Speed : REAL;
        StartTime : REAL;
        StopTime : REAL;

        Cargo : LoadQObj;


        {Methods}

         {Vehicle ID Data}


   ASK METHOD SetVehicleID(IN NewVehicleID : INTEGER);
   ASK METHOD InputClass;
   ASK METHOD SetClass(IN NewClass : STRING);
   ASK METHOD InputSubClass;
   ASK METHOD SetSubClass(IN NewSubClass : STRING);
   ASK METHOD SetOverSize(IN NewSetOverSize : STRING);

        {Vehicle Performance Data}

   ASK METHOD SetLength(IN NewLength : REAL);
   ASK METHOD SetWidth(IN NewWidth : REAL);
   ASK METHOD SetMaxSpeed(IN NewMaxSpeed : REAL);
   ASK METHOD SetMaxRange(IN NewMaxRange : REAL);
   ASK METHOD SetMaxCargoCube(IN NewMaxCargoCube : REAL);
   ASK METHOD SetMaxCargoArea(IN NewMaxCargoArea : REAL);
   ASK METHOD SetMaxCargoWeight(IN NewMaxCargoWeight : REAL);
   ASK METHOD SetMaxCargoLength(IN NewMaxCargoLength : REAL);
   ASK METHOD SetMaxCargoWidth(IN NewMaxCargoWidth : REAL);
   ASK METHOD SetMaxCargoHeight(IN NewMaxCargoHeight : REAL);
```

```
ASK METHOD SetMaxPax(IN NewMaxPax : REAL);ASK METHOD SetMaxGas(IN NewMaxGas : RE

     {Movement Data}

ASK METHOD SetLocation(INOUT NewLocation : BaseObj);
ASK METHOD SetPosition(IN NewPosition : PositionRecType);
ASK METHOD SetDestination(INOUT NewDestination : BaseObj);
ASK METHOD SetStatus(IN NewStatus : MovementStatusType);
ASK METHOD SetPort(INOUT Base : BaseObj);

ASK METHOD SetSpeed(IN NewSpeed : REAL);
ASK METHOD SetStartTime(IN NewStartTime : REAL);

ASK METHOD LoadOut(INOUT Load : LoadQObj);
ASK METHOD GiveCurrentPosition() : PositionRecType;

ASK METHOD Arrive;
TELL METHOD GoTo(IN NewNode : BaseObj);
TELL METHOD Unload;

ASK METHOD Modify;
ASK METHOD Display;

ASK METHOD ObjInit;
ASK METHOD CleanUp;

OVERRIDE

ASK METHOD ObjTerminate;

END OBJECT;

{=========================================================================}
TransporterQObj = PROTO(MyQueueObj[NamedObj : #TransporterObj]);
{=========================================================================}
OVERRIDE

ASK METHOD Display(INOUT j : INTEGER);


END PROTO;

END MODULE.
```

```
IMPLEMENTATION MODULE Trnsprt;

{Comments}

{Import statements}
FROM Base IMPORT BaseObj;
FROM Port IMPORT PortObj;
FROM CommodQ IMPORT ALL CommodityClassType,
                    CommodityObj,
                    CommodityQObj;
FROM Distant IMPORT PositionRecType,
                    OutputPosition;
FROM Node IMPORT NodeObj;
FROM MyQueue IMPORT MyQueueObj;
FROM Distant IMPORT CalcDistance;
FROM SimMod IMPORT SimTime,
                   InterruptAll;
FROM TManage IMPORT TransporterManager;
FROM WriteLine IMPORT WriteLine;
FROM CRTMod IMPORT ClearScreen;
FROM Builder IMPORT Builder;
FROM Trash IMPORT TrashCan;
FROM SOUTPUT IMPORT SOUTPUT;
FROM Shpmnt IMPORT ShipmentObj,
                   ShipmentQObj;
FROM IOMod IMPORT ReadKey;
{Definitions}

{==========================================================================}
OBJECT LoadQObj;
{==========================================================================}

        {METHODS}

{--------------------------------------------------------------------------}
ASK METHOD Add(IN NewMember : ShipmentObj);
{--------------------------------------------------------------------------}

VAR
BEGIN

INHERITED Add(NewMember);

CASE NewMember.Item.Class
WHEN Personnel:
        TotalPax := TotalPax + NewMember.Item.OnHand;
WHEN Fuel:
        TotalGas := TotalGas + NewMember.Item.OnHand;
WHEN Major:
        TotalArea := TotalArea + (NewMember.Item.OnHand * NewMember.Item.Length
                                  * NewMember.Item.Width )/ 144.00;
        TotalCube := TotalCube + (NewMember.Item.OnHand * NewMember.Item.Length
                  * NewMember.Item.Width * PalletHeight)/1728.00;


OTHERWISE
        TotalCube := TotalCube + (NewMember.Item.OnHand * NewMember.Item.Length
                  * NewMember.Item.Width * NewMember.Item.Height)/1728.00;
        TotalArea := TotalArea + ((NewMember.Item.OnHand * NewMember.Item.Length
                  * NewMember.Item.Width * NewMember.Item.Height) / PalletHeight)
                  / 144.00;
         IF NewMember.Item.OverSize
                  TotalOutSize := TotalOutSize + NewMember.Item.OnHand;
         END IF;
END CASE;
TotalWeight := TotalWeight + (NewMember.Item.OnHand * NewMember.Item.Weight);

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD Remove : ShipmentObj;
{----------------------------------------------------------------------------}

VAR
Shipment : ShipmentObj;

BEGIN

Shipment := INHERITED Remove;

CASE Shipment.Item.Class
WHEN Personnel:
         TotalPax := TotalPax - Shipment.Item.OnHand;
WHEN Fuel:
         TotalGas := TotalGas - Shipment.Item.OnHand;
WHEN Major:
         TotalArea := TotalArea - (Shipment.Item.OnHand * Shipment.Item.Length
                   * Shipment.Item.Width) / 144.00;
         TotalCube := TotalCube - (Shipment.Item.OnHand * Shipment.Item.Length
                   * Shipment.Item.Width * PalletHeight)/1728.00;
OTHERWISE
         TotalCube := TotalCube - (Shipment.Item.OnHand * Shipment.Item.Length *
                  Shipment.Item.Width * Shipment.Item.Height)/1728.00;
         TotalArea := TotalArea - ((Shipment.Item.OnHand * Shipment.Item.Length
                   * Shipment.Item.Width * Shipment.Item.Height) / PalletHeight)
                   / 144.00;
         IF Shipment.Item.OverSize
                  TotalOutSize := TotalOutSize - Shipment.Item.OnHand;
         END IF;

END CASE;
TotalWeight := TotalWeight - (Shipment.Item.OnHand * Shipment.Item.Weight);

RETURN(Shipment);

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD RemoveThis(IN member : ShipmentObj);
{----------------------------------------------------------------------------}

VAR
BEGIN

IF ASK SELF Includes(member)

         CASE member.Item.Class
         WHEN Personnel:
                  TotalPax := TotalPax - member.Item.OnHand;
         WHEN Fuel:
                TotalGas := TotalGas - member.Item.OnHand;
        WHEN Major:
                TotalArea := TotalArea - (member.Item.OnHand *
                         member.Item.Length * member.Item.Width) / 144.00;
                TotalCube := TotalCube - (member.Item.OnHand *
                         member.Item.Length * member.Item.Width *
                         PalletHeight)/1728.00;
        OTHERWISE
                TotalCube := TotalCube - (member.Item.OnHand *
                         member.Item.Length * member.Item.Width *
                         member.Item.Height) / 1728.00;
                TotalArea := TotalArea - ((member.Item.OnHand *
                         member.Item.Length * member.Item.Width *
                         member.Item.Height) / PalletHeight) / 144.00;
                IF member.Item.OverSize
                        TotalOutSize := TotalOutSize - member.Item.OnHand;
                END IF;
END CASE;
TotalWeight := TotalWeight - (member.Item.OnHand * member.Item.Weight);

END IF;

INHERITED RemoveThis(member);

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetTotalCube(IN NewTotalCube : REAL);
{-----------------------------------------------------------------------}

VAR
BEGIN

TotalCube := NewTotalCube;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetTotalArea(IN NewTotalArea : REAL);
{-----------------------------------------------------------------------}

VAR
BEGIN

TotalCube := NewTotalArea;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetTotalWeight(IN NewTotalWeight : REAL);
{-----------------------------------------------------------------------}

VAR
BEGIN

TotalWeight := NewTotalWeight.

END METHOD;

{-----------------------------------------------------------------------}
```

```
ASK METHOD SetTotalPax(IN NewTotalPax : REAL);
{----------------------------------------------------------------------}

VAR
BEGIN

TotalPax := NewTotalPax;

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD SetTotalGas(IN NewTotalGas : REAL);
{----------------------------------------------------------------------}

VAR
BEGIN

TotalGas := NewTotalGas;

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD SetTotalOutSize(IN NewTotalOutSize : REAL);
{----------------------------------------------------------------------}

VAR
BEGIN

TotalOutSize := NewTotalOutSize;

END METHOD;

END OBJECT;

{======================================================================}
OBJECT TransporterObj;
{======================================================================}

{METHODS}

{----------------------------------------------------------------------}
ASK METHOD SetVehicleID(IN NewVehicleID : INTEGER);
{----------------------------------------------------------------------}

VAR

BEGIN

VehicleID := NewVehicleID;

END METHOD;

{----------------------------------------------------------------------}
ASK METHOD InputClass;
{----------------------------------------------------------------------}

VAR
CHR : CHAR;
BEGIN
```

```
LOOP
        OUTPUT("     Input Transporter Class:  ");
        OUTPUT("     (A)ir, (R)ail, (S)hip, (T)ruck");
        CHR := ReadKey();
        IF (CHR = "A") OR (CHR = "a")
                SetClass("Aircraft");
                EXIT;
        ELSIF  (CHR = "R") OR (CHR = "r")
                SetClass("Rail");
                EXIT;
        ELSIF  (CHR = "S") OR (CHR = "s")
                SetClass("Ship");
                EXIT;
        ELSIF  (CHR = "T") OR (CHR = "t")
                SetClass("Truck");
                EXIT;
        END IF;
END LOOP;
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetClass(IN NewClass : STRING);
{---------------------------------------------------------------------------}

VAR

BEGIN

CASE NewClass
        WHEN "Aircraft":
                Class := Aircraft;
        WHEN "Ship":
                Class := Ship;
        WHEN "Rail":
                Class := Rail;
        WHEN "Truck":
                Class := Truck;
        OTHERWISE
                OUTPUT("Transporter class out of range - SetClass");
END CASE;

END METHOD;


{---------------------------------------------------------------------------}
ASK METHOD InputSubClass;
{---------------------------------------------------------------------------}

VAR
CHR : CHAR;

BEGIN

LOOP
        OUTPUT("     Input Transporter Class:  ");
        OUTPUT("     (P)ax, (L)iquid, (R)oRo, (G)eneral, (B)reakBulk");
        CHR := ReadKey();
        IF  (CHR = "P") OR (CHR = "p")
                SetSubClass("Pax");
                EXIT;
        ELSIF  (CHR = "L") OR (CHR = "l")
                SetSubClass("Liquid");
                EXIT;
        ELSIF  (CHR = "R") OR (CHR = "r")
                SetSubClass("RoRo");
                EXIT;
        ELSIF  (CHR = "B") OR (CHR = "b")
                SetSubClass("BreakBulk");
                EXIT;
        ELSIF  (CHR = "G") OR (CHR = "g")
                SetSubClass("General");
                EXIT;
        END IF;
END LOOP;
END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetSubClass(IN NewSubClass : STRING);
{---------------------------------------------------------------------------}

VAR

BEGIN

CASE NewSubClass
        WHEN "Liquid":
                SubClass := Liquid;
        WHEN "RoRo":
                SubClass := RoRo;
        WHEN "BreakBulk":
                SubClass := BreakBulk;
        WHEN "General":
                SubClass := General;
        WHEN "Pax":
                SubClass := Pax;
        OTHERWISE
                OUTPUT("Transporter subclass out of range - SetSubClass");
END CASE;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetOverSize(IN NewSetOverSize : STRING);
{---------------------------------------------------------------------------}

VAR

BEGIN

IF NewSetOverSize = "TRUE"
        OverSize := TRUE;
ELSIF NewSetOverSize = "FALSE"
        OverSize := FALSE;
ELSE
        OUTPUT("Transporter OutSize - OUT OF RANGE");
END IF;

END METHOD;
```

```
     {Vehicle Performance Data}
{-----------------------------------------------------------------------------}
ASK METHOD SetLength(IN NewLength : REAL);
{-----------------------------------------------------------------------------}

VAR

BEGIN

Length := NewLength;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SetWidth(IN NewWidth : REAL);
{-----------------------------------------------------------------------------}

VAR

BEGIN

Width := NewWidth;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SetMaxSpeed(IN NewMaxSpeed : REAL);
{-----------------------------------------------------------------------------}

VAR

BEGIN

MaxSpeed := NewMaxSpeed;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SetMaxRange(IN NewMaxRange : REAL);
{-----------------------------------------------------------------------------}

VAR

BEGIN

MaxRange := NewMaxRange;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SetMaxCargoCube(IN NewMaxCargoCube : REAL);
{-----------------------------------------------------------------------------}

VAR

BEGIN

MaxCargoCube := NewMaxCargoCube;

END METHOD;
```

```
{---------------------------------------------------------------------}
ASK METHOD SetMaxCargoArea(IN NewMaxCargoArea : REAL);
{---------------------------------------------------------------------}

VAR

BEGIN

MaxCargoArea := NewMaxCargoArea;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD SetMaxCargoWeight(IN NewMaxCargoWeight : REAL);
{---------------------------------------------------------------------}

VAR

BEGIN

MaxCargoWeight := NewMaxCargoWeight;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD SetMaxCargoLength(IN NewMaxCargoLength : REAL);
{---------------------------------------------------------------------}

VAR

BEGIN

MaxCargoLength := NewMaxCargoLength;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD SetMaxCargoWidth(IN NewMaxCargoWidth : REAL);
{---------------------------------------------------------------------}

VAR

BEGIN

MaxCargoWidth := NewMaxCargoWidth;

END METHOD;

{---------------------------------------------------------------------}
ASK METHOD SetMaxCargoHeight(IN NewMaxCargoHeight : REAL);
{---------------------------------------------------------------------}

VAR

BEGIN

MaxCargoHeight := NewMaxCargoHeight;

END METHOD;
```

```
{----------------------------------------------------------------------------}
ASK METHOD SetMaxPax(IN NewMaxPax : REAL);
{----------------------------------------------------------------------------}

VAR

BEGIN

MaxPax := NewMaxPax;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD SetMaxGas(IN NewMaxGas : REAL);
{----------------------------------------------------------------------------}

VAR

BEGIN

MaxGas := NewMaxGas;

END METHOD;

     {movement Data}
{----------------------------------------------------------------------------}
ASK METHOD SetLocation(INOUT NewLocation : BaseObj);
{----------------------------------------------------------------------------}

VAR

BEGIN

Location := NewLocation;

END METHOD;

{----------------------------------------------------------------------------}
ASK METHOD SetPosition(IN NewPosition : PositionRecType);
{----------------------------------------------------------------------------}

VAR

BEGIN

Position := NewPosition;

END METHOD;


{----------------------------------------------------------------------------}
ASK METHOD SetDestination(INOUT NewDestination : BaseObj);
{----------------------------------------------------------------------------}

VAR

BEGIN

Destination := NewDestination;
```

```
END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetStatus(IN NewStatus : MovementStatusType);
{-----------------------------------------------------------------------}

VAR

BEGIN

Status := NewStatus;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetPort(INOUT Base: BaseObj);
{-----------------------------------------------------------------------}

VAR

BEGIN
          CASE Class
                WHEN Aircraft :
                        Port := ASK Base AirPort;
                WHEN Ship :
                        Port := ASK Base SeaPort;
                WHEN Truck :
                        Port := ASK Base TruckStop;
                WHEN Rail :
                        Port := ASK Base RailYard;

                OTHERWISE
                        OUTPUT("No Proper Port - AM SetPort");
                        HALT;
          END CASE;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetSpeed(IN NewSpeed : REAL);
{-----------------------------------------------------------------------}

VAR

BEGIN

Speed := NewSpeed;

END METHOD;

{-----------------------------------------------------------------------}
ASK METHOD SetStartTime(IN NewStartTime : REAL);
{-----------------------------------------------------------------------}

VAR

BEGIN

StartTime := NewStartTime;
```

```
END METHOD;

{-----------------------------------------------------------------------}
     ASK METHOD LoadOut(INOUT Load : LoadQObj);
{-----------------------------------------------------------------------}

{THE ACTUAL MEANS OF TAKING ON CARGO}

VAR

i,numItems : INTEGER;
CurrentItem : ShipmentObj;

BEGIN

{
OUTPUT("IN LoadOut - ", Name, VehicleID);
}
ASK SELF TO SetStatus(LOADING);
numItems := ASK Load numberIn;
FOR i := 1 TO numItems
        CurrentItem := ASK Load TO Remove;
        ASK Cargo TO Add(CurrentItem);
END FOR;

END METHOD;

{-----------------------------------------------------------------------}
     ASK METHOD GiveCurrentPosition() : PositionRecType;
{-----------------------------------------------------------------------}
     BEGIN
        RETURN(Position);
     END METHOD;

{-----------------------------------------------------------------------}
     ASK METHOD Arrive;
{-----------------------------------------------------------------------}
     VAR
     BEGIN
{
OUTPUT("IN Arrive - ", Name, VehicleID);
}
{WriteLine(Name + INTTOSTR(VehicleID) + " arriving at " + Destination.Name);}
          Location := Destination;
          Position := ASK Destination Position;
          ASK SELF TO SetPort(Destination);

          ASK Port TO GetArrival(SELF);
     END METHOD;

{-----------------------------------------------------------------------}
     TELL METHOD GoTo(IN NewNode : BaseObj);
{-----------------------------------------------------------------------}
VAR

NewWayPoint : PositionRecType;
Distance : REAL;
TravelTime : REAL;
LoadTime : REAL;
```

```
BEGIN

OUTPUT("IN GoTo - ", NewNode.Name,", ", Name, VehicleID);

                                                        {Calculate Loading time}

CASE Class
WHEN Aircraft:
        LoadTime := 4.0 * (Cargo.TotalCube/MaxCargoCube);
WHEN Ship:
        CASE SubClass
        WHEN RoRo:
                LoadTime := 48.0 * (Cargo.TotalCube/MaxCargoCube);
        WHEN Liquid:
                LoadTime := 24.0 * (Cargo.TotalGas/MaxGas);
        OTHERWISE
                LoadTime := 96.0 * (Cargo.TotalCube/MaxCargoCube);
        END CASE;
WHEN Truck:
        CASE SubClass
        WHEN Liquid:
                LoadTime := 2.0 * (Cargo.TotalGas/MaxGas);
        OTHERWISE
                LoadTime := 2.0 * (Cargo.TotalCube/MaxCargoCube);
        END CASE;
WHEN Rail:
        LoadTime := 2.0;
END CASE;
ASK SELF TO SetStatus(LOADING);
IF LoadTime < 0.0
        LoadTime := 0.0;
END IF;
StartTime := SimTime();
StopTime := SimTime() + LoadTime;
                                                        {Calculate Travel time}

Destination := NewNode;
NewWayPoint := ASK Destination Position;
Distance := CalcDistance(Position, NewWayPoint);

ASK SELF TO SetSpeed(MaxSpeed);                         {change for variable speeds}
IF Speed > 0.0
        TravelTime := Distance / Speed;
ELSE
        OUTPUT("Transporter Speed <= 0");
        HALT;
END IF;

OUTPUT("Load Time is ", LoadTime, " Travel Time is ", TravelTime);

WAIT DURATION LoadTime

        IF Port <> NILOBJ
                ASK Port TO GetDeparture(SELF);
        END IF;
        ASK SELF TO SetStatus(ENROUTE);
        StartTime := SimTime();
        StopTime := SimTime() + TravelTime;
        WAIT DURATION TravelTime;
                ASK SELF TO Arrive;
        ON INTERRUPT
        END WAIT;

ON INTERRUPT
END WAIT

END METHOD;

{----------------------------------------------------------------------}
TELL METHOD Unload;
{----------------------------------------------------------------------}
VAR

UnloadTime : REAL:

BEGIN
{
OUTPUT("IN UnLoad - ",Name, VehicleID);
}
ASK SELF TO SetStatus(UNLOADING);

                                                {Calculate Unloading Time}

CASE Class
WHEN Aircraft:
        UnloadTime := 4.0 * (Cargo.TotalCube/MaxCargoCube);
WHEN Ship:
        CASE SubClass
        WHEN RoRo:
                UnloadTime := 48.0 * (Cargo.TotalCube/MaxCargoCube);
        WHEN Liquid:
                UnloadTime := 24.0 * (Cargo.TotalGas/MaxGas);
        OTHERWISE
                UnloadTime := 96.0 * (Cargo.TotalCube/MaxCargoCube);
        END CASE;
WHEN Truck:
        CASE SubClass
        WHEN Liquid:
                UnloadTime := 2.0 * (Cargo.TotalGas/MaxGas);
        OTHERWISE
                UnloadTime := 2.0 * (Cargo.TotalCube/MaxCargoCube);
        END CASE;
WHEN Rail:
        UnloadTime := 1.0;
END CASE;
IF UnloadTime < 0.0
        UnloadTime := 0.0;
END IF;
StartTime := SimTime();
StopTime := SimTime() + UnloadTime;

                                                {Wait Duration based on Cube}
WAIT DURATION UnloadTime
        ASK Destination TO ReceiveStuff(Cargo);
        ASK Cargo TO SetTotalWeight(0.0);
        ASK Cargo TO SetTotalCube(0.0);
        ASK Port TO Load(SELF);
```

```
ON INTERRUPT
END WAIT;

END METHOD;                                                  {WRITE THIS !!!!!}

{--------------------------------------------------------------------------}
     ASK METHOD Display;
{--------------------------------------------------------------------------}

CONST

format = "      ***.  **************** ********** **";
format2 = "      *****************  *********   *****************  **********";
format4 =       "                ****************  *** *** *";
title = "==================== ***************>-*** at time ******<  ============

VAR

j, i, numItems : INTEGER;
item : CommodityObj;
string, Answer : STRING;
Shipment : ShipmentObj;

BEGIN

ClearScreen;
SOUTPUT(" ", j);
string := SPRINT(Name,VehicleID,SimTime) WITH title;
SOUTPUT(string, j);
SOUTPUT(" ", j);

OUTPUT("      Class:  ", Class);
OUTPUT("      SubClass:  ", SubClass);
OUTPUT("      OutSize:  " ,OverSize);
j := j + 3;

IF Position <> NILREC
        SOUTPUT(" ", j);
        OutputPosition(Position, j);
        SOUTPUT(" ", j);
END IF;

IF Location <> NILOBJ
        SOUTPUT("      Location:   " + Location.Name, j);
END IF;
IF Destination <> NILOBJ
        SOUTPUT("      Destination:   " + Destination.Name, j);
END IF;
        OUTPUT("      Status:  ", Status);
        j := j + 1;
        SOUTPUT("      Start:   " + REALTOSTR(StartTime), j);
        SOUTPUT("      Stop:   " + REALTOSTR(StopTime), j);
        SOUTPUT(" ",j);

string := SPRINT("Length:", TRUNC(Length), "Width:", TRUNC(Width))
         WITH format2;
SOUTPUT(string, j);
string := SPRINT("Max Speed:", TRUNC(MaxSpeed), "Max Range:",
         TRUNC(MaxRange)) WITH format2;
SOUTPUT(string, j);
SOUTPUT(" ", j);

string := SPRINT("Max Area:", TRUNC(MaxCargoArea), "Max Cube:",
        TRUNC(MaxCargoCube)) WITH format2;
SOUTPUT(string, j);
string := SPRINT("Max Weight:", TRUNC(MaxCargoWeight), " ", " ")
        WITH format2;
SOUTPUT(string, j);
string := SPRINT("Max Item Length:", TRUNC(MaxCargoLength),"Max Item Width:",
        TRUNC(MaxCargoWidth)) WITH format2;
SOUTPUT(string, j);
string := SPRINT("Max Item Height:", TRUNC(MaxCargoHeight), " ", " ")
        WITH format2;
SOUTPUT(string, j);
string := SPRINT("Max Num Pax:", TRUNC(MaxPax), "Max Liquid:",
        TRUNC(MaxGas)) WITH format2;
SOUTPUT(string, j);
SOUTPUT(" ", j);

SOUTPUT("     Cargo: ", j);
SOUTPUT(" ", j);

numItems := ASK Cargo numberIn;
IF numItems = 0
        SOUTPUT("          EMPTY", j);
ELSE
        FOR i := 1 TO numItems
                Shipment := ASK Cargo TO Remove;
                ASK Cargo TO Add(Shipment);
                string := SPRINT(i, Shipment.Item.Name, Shipment.Item.OnHand,
                          Shipment.Item.Priority) WITH format;
                SOUTPUT(string, j);
        END FOR;
END IF;
SOUTPUT(" ", j);
SOUTPUT("====================================================================

SOUTPUT(" ", j);


END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD Modify;
{---------------------------------------------------------------------------}
VAR

CHR : CHAR;
string : STRING;
real : REAL;
j, integer : INTEGER;


BEGIN

LOOP
ClearScreen;
ASK SELF TO Display;
OUTPUT("");
OUTPUT("     Modify? (N)ame, (D)imensions, (P)erformance, Car(g)o," +
" (R)eturn.");
CHR := ReadKey();

IF (CHR = "N") OR (CHR = "n")
        ASK SELF TO InputName;

ELSIF (CHR = "D") OR (CHR = "d")

        OUTPUT("     Current Length is ", Length);
        OUTPUT("     Input  new Transporter Length (REAL feet)");
        INPUT(real);
        ASK SELF TO SetLength(real);

        OUTPUT("     Current Length is ", Width);
        OUTPUT("     Input new Transporter Width (REAL feet)");
        INPUT(real);
        ASK SELF TO SetWidth(real);

ELSIF (CHR = "P") OR (CHR = "p")

        OUTPUT("     Current Speed is ", MaxSpeed);
        OUTPUT("     Input  new Transporter Speed (REAL feet)");
        INPUT(real);
        ASK SELF TO SetMaxSpeed(real);

        OUTPUT("     Current Range is ", MaxRange);
        OUTPUT("     Input new Transporter Range (REAL feet)");
        INPUT(real);
        ASK SELF TO SetMaxRange(real);

ELSIF (CHR = "G") OR (CHR = "g")
        LOOP

        OUTPUT("     (L)ength, (W)idth, (H)eight, (C)ube, (A)rea,");
        OUTPUT("     Wei(g)ht, (P)assengers, Li(q)uids, (R)eturn.");
        CHR := ReadKey();
        IF (CHR = "l") OR (CHR = "L")
                OUTPUT("     Largest Single Item Length is ", MaxCargoLength);
                OUTPUT("     Input new Length (REAL inches).");
                INPUT(real);
                ASK SELF TO SetMaxCargoLength(real);

        ELSIF (CHR = "w") OR (CHR = "W")
                OUTPUT("     Largest Single Item Width is ", MaxCargoWidth);
                OUTPUT("     Input new Width (REAL inches).");
                INPUT(real);
                ASK SELF TO SetMaxCargoWidth(real);
        ELSIF (CHR = "h") OR (CHR = "H")
                OUTPUT("     Largest Single Item Height is ", MaxCargoHeight);
                OUTPUT("     Input new Height (REAL inches).");
                INPUT(real);
                ASK SELF TO SetMaxCargoHeight(real);
        ELSIF (CHR = "c") OR (CHR = "C")
                OUTPUT("     Maximum Cargo Cube is ", MaxCargoCube);
                OUTPUT("     Input new Maximum Cargo Cube (REAL cu. ft.).");
                INPUT(real);
                ASK SELF TO SetMaxCargoCube(real);
        ELSIF (CHR = "a") OR (CHR = "A")
```

```
                OUTPUT("     Maximum Cargo Area is ", MaxCargoArea);
                OUTPUT("     Input new Maximum Cargo Area (REAL sq. ft.).");
                INPUT(real);
                ASK SELF TO SetMaxCargoArea(real);
        ELSIF (CHR = "g") OR (CHR = "G")
                OUTPUT("     Maximum Cargo Weight is ", MaxCargoWeight);
                OUTPUT("     Input new Maximum Cargo Weight (REAL lbs.).");
                INPUT(real);
                ASK SELF TO SetMaxCargoWeight(real);
        ELSIF (CHR = "p") OR (CHR = "P")
                OUTPUT("     Maximum number of Passengers is ", MaxPax);
                OUTPUT("     Input new Passengers capacity.");
                INPUT(real);
                ASK SELF TO SetMaxPax(real);
        ELSIF (CHR = "q") OR (CHR = "Q")
                OUTPUT("     Liquid Cargo Capacity is ", MaxGas);
                OUTPUT("     Input new Liquid Cargo Capacity (REAL bbls.).");
                INPUT(real);
                ASK SELF TO SetMaxGas(real);
        ELSIF (CHR = "r") OR (CHR = "R")
                EXIT;
        END IF;

        END LOOP;

ELSIF (CHR = "R") OR (CHR = "r")
        EXIT;
END IF;

END LOOP;
END METHOD;


{---------------------------------------------------------------------------}
     ASK METHOD ObjInit;
{---------------------------------------------------------------------------}

VAR

BEGIN

NEW(Cargo);
{
NEW(Position);
Position.LatDeg := 0;
Position.LatMin := 0;
Position.LatDir := "N";
Position.LongDeg := 0;
Position.LongMin := 0;
Position.LongDir := "E";
}

END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD ObjTerminate;
{---------------------------------------------------------------------------}

VAR
```

```
BEGIN

DISPOSE(Cargo);

END METHOD;

{-------------------------------------------------------------------------}
     ASK METHOD CleanUp;
{-------------------------------------------------------------------------}

VAR

i, numItems : INTEGER;
commodity,commodity2 : CommodityObj;
Shipment : ShipmentObj;

BEGIN

InterruptAll(SELF);
CASE Class
WHEN Aircraft:
        IF ASK TransporterManager.AvailableAircraft TO Includes(SELF)
                ASK TransporterManager.AvailableAircraft TO RemoveThis(SELF);
        END IF;
WHEN Rail:
        IF ASK TransporterManager.AvailableTrains TO Includes(SELF)
                ASK TransporterManager.AvailableTrains TO RemoveThis(SELF);
        END IF;
WHEN Truck:
        IF ASK TransporterManager.AvailableTrucks TO Includes(SELF)
                ASK TransporterManager.AvailableTrucks TO RemoveThis(SELF);
        END IF;
WHEN Ship:
        IF ASK TransporterManager.AvailableShips TO Includes(SELF)
                ASK TransporterManager.AvailableShips TO RemoveThis(SELF);
        END IF;
END CASE;
IF Builder <> NILOBJ
        IF ASK Builder.BigTransporterQ TO Includes(SELF);
                ASK Builder.BigTransporterQ TO RemoveThis(SELF);
        END IF;
END IF;
IF ASK Port.ArrivalsQ TO Includes(SELF);
        ASK Port.ArrivalsQ TO RemoveThis(SELF);
END IF;

IF ASK Port.BerthsQ TO Includes(SELF);
        ASK Port.BerthsQ TO RemoveThis(SELF);
END IF;

IF ASK Port.ParkedQ TO Includes(SELF);
        ASK Port.ArrivalsQ TO RemoveThis(SELF);
END IF;

numItems := ASK Cargo numberIn;
FOR i := 1 TO numItems
        Shipment := ASK Cargo TO Remove;
        commodity := ASK Shipment Item;
        commodity2 := ASK Destination.Inventory TO FindByName(commodity.Name);
        ASK commodity2 TO SubtractOnOrder(commodity.OnHand);
END FOR;

ASK TrashCan TO Add(SELF);

END METHOD;

END OBJECT;

{============================================================================}
PROTO TransporterQObj;
{============================================================================}

{----------------------------------------------------------------------------}
ASK METHOD Display(INOUT j :INTEGER);
{----------------------------------------------------------------------------}

CONST

format = "     ***************     ***************     *************** ";

VAR

string, string1, string2, string3 : STRING;
transporter, lasttransporter : TransporterObj;
name : STRING;
ID : INTEGER;

BEGIN
SOUTPUT(" ",j);

IF ASK SELF numberIn > 0
        lasttransporter := ASK  SELF Last;
        LOOP
                string1 := "               ";
                string2 := "               ";
                string3 := "               ";

                transporter := ASK SELF TO Remove;
                ASK SELF TO Add(transporter);
                name := ASK transporter Name;
                ID := ASK transporter VehicleID;
                string1 := name + "#" + INTTOSTR(ID);
                IF transporter = lasttransporter
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;

                transporter := ASK SELF TO Remove;
                ASK SELF TO Add(transporter);
                name := ASK transporter Name;
                ID := ASK transporter VehicleID;
                string2 := name + "#" + INTTOSTR(ID) ;
                IF transporter = lasttransporter
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;
                transporter := ASK SELF TO Remove;
                ASK SELF TO Add(transporter );
                name  := ASK transporter  Name;
                ID  := ASK transporter  VehicleID;
                string3 := name + "#" + INTTOSTR(ID) ;
                IF transporter  = lasttransporter
                        string := SPRINT(string1,string2,string3) WITH format;
                        SOUTPUT(string,j);
                        EXIT;
                END IF;
                string := SPRINT(string1,string2,string3) WITH format;
                SOUTPUT(string,j);

        END LOOP;
END IF;

END METHOD;

END PROTO;


END MODULE.
```

```
DEFINITION MODULE Unit;

{Basic Unit Object}

{Import statements}

FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj,
                    ALL CommodityClassType;

FROM Node IMPORT NodeObj;
FROM Distant IMPORT PositionRecType;
FROM Base IMPORT BaseObj,
                 BaseQObj;
FROM Builder IMPORT BuilderObj;
FROM MyQueue IMPORT MyQueueObj,
                    NamedObj;
FROM Shpmnt IMPORT ShipmentObj;

{Type Declarations}

TYPE

UnitClassType = (Air, Sea, Land);

CombatIntensityType = (High, Med, Low, None);

{===========================================================================}
UnitObj = OBJECT(BaseObj)
{===========================================================================}

     {Fields}

Class : UnitClassType;
InPlace : BOOLEAN;
ActiveAt : REAL;
CombatIntensity : CombatIntensityType;
Origin : BaseObj;
Linked : BOOLEAN;
DelayUntil : REAL;

     {Methods}

ASK METHOD SetClass(IN NewClass : STRING);
ASK METHOD SetOrigin(INOUT NewOrigin : BaseObj);
ASK METHOD SetInPlace(IN NewInPlace : STRING);
ASK METHOD SetActiveAt(IN NewActiveAt : REAL);
ASK METHOD SetCombatIntensity(IN NewCombatIntensity : STRING);
ASK METHOD SetLinked;
ASK METHOD SetDelayUntil(IN NewDelayUntil : REAL);
ASK METHOD ChangePosition(IN NewPosition : PositionRecType);
ASK METHOD Activate;
ASK METHOD InputDelayUntil;
TELL METHOD Consume;
TELL METHOD DelayActivation;


     OVERRIDE

ASK METHOD Display;
ASK METHOD Modify(INOUT builder : BuilderObj);
ASK METHOD OrderStuff(INOUT Item : CommodityObj);
ASK METHOD FillOrder(INOUT Shipment : ShipmentObj);
ASK METHOD BackOrderStuff(INOUT BackOrderShipment : ShipmentObj);
ASK METHOD FillBackOrders;
TELL METHOD CheckInventory;

ASK METHOD ObjInit;

END OBJECT;

{========================================================================}
UnitQObj = PROTO(BaseQObj[BaseObj : #UnitObj])
{========================================================================}
END PROTO;


END MODULE.
```

```
IMPLEMENTATION MODULE Unit;

{Generic Unit Object}

{Import statements}

FROM Distant IMPORT PositionRecType,
                    InputPosition,
                    OutputPosition;
FROM Node IMPORT NodeObj;

FROM CommodQ IMPORT CommodityObj,
                    CommodityQObj,
                    ALL CommodityClassType;
FROM Trnsprt IMPORT ALL TransporterClassType,
                        TransporterObj,
                        TransporterQObj;
FROM Base IMPORT BaseObj;
FROM Port IMPORT PortObj;
FROM Builder IMPORT BuilderObj,
                    Builder;

FROM SimMod IMPORT SimTime,
                   Interrupt;
FROM IOMod IMPORT ReadKey;
FROM CRTMod IMPORT ClearScreen;
FROM WriteLine IMPORT WriteLine;
FROM SOUTPUT IMPORT SOUTPUT;
FROM Shpmnt IMPORT ShipmentObj;
FROM LogMan IMPORT LogisticsManager;
FROM ScenEd IMPORT ScenarioEditor;
FROM TManage IMPORT TransporterManager;
{Definitions}

{===========================================================================}
OBJECT UnitObj;
{===========================================================================}

     {METHODS}
{---------------------------------------------------------------------------}
ASK METHOD SetClass(IN NewClass : STRING);
{---------------------------------------------------------------------------}

BEGIN

CASE NewClass

        WHEN "Air":
                Class := Air;
        WHEN "Sea":
                Class := Sea;
        WHEN "Land":
                Class := Land;
        OTHERWISE
                Class := Sea;
END CASE;

END METHOD;

{---------------------------------------------------------------------------}
```

```
ASK METHOD SetOrigin(INOUT NewOrigin : BaseObj);
{---------------------------------------------------------------------------}

BEGIN

Origin := NewOrigin;

END METHOD;
{---------------------------------------------------------------------------}
ASK METHOD SetInPlace(IN NewInPlace : STRING);
{---------------------------------------------------------------------------}

BEGIN

IF NewInPlace = "TRUE"
        InPlace := TRUE;
ELSE
        InPlace := FALSE;
END IF;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetActiveAt(IN NewActiveAt : REAL);
{---------------------------------------------------------------------------}

BEGIN

ActiveAt := NewActiveAt;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetCombatIntensity(IN NewCombatIntensity : STRING);
{---------------------------------------------------------------------------}

BEGIN

CASE NewCombatIntensity
        WHEN "High" :
                CombatIntensity := High;
        WHEN "Med" :
                CombatIntensity := Med;
        WHEN "Low" :
                CombatIntensity := Low;
        WHEN "None" :
                CombatIntensity := None;
        OTHERWISE
                OUTPUT("Invalid Combat Intensity Factor");
                HALT;
END CASE;

END METHOD;

{---------------------------------------------------------------------------}
ASK METHOD SetLinked;
{---------------------------------------------------------------------------}

BEGIN
```

```
Linked := TRUE;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD SetDelayUntil(IN NewDelayUntil : REAL); {---------------------------

BEGIN

DelayUntil :=  NewDelayUntil;

END METHOD;

{-----------------------------------------------------------------------------}
     ASK METHOD ChangePosition(IN NewPosition : PositionRecType);
{-----------------------------------------------------------------------------}
     BEGIN
          Position := NewPosition;
     END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD Activate;
{-----------------------------------------------------------------------------}

BEGIN

Interrupt(SELF, "DelayActivation");
TELL SELF TO Consume;

END METHOD;

{-----------------------------------------------------------------------------}
ASK METHOD InputDelayUntil;
{-----------------------------------------------------------------------------}

{INTERACTIVELY SETS UNIT DELAYUNTIL FIELD}

VAR

CHR : CHAR;
real : REAL;

BEGIN

LOOP
        OUTPUT("     Do you wish the unit to delay activation? (Y or N)");
        CHR := ReadKey();

        IF (CHR = "Y") OR (CHR = "y")
                OUTPUT("     Input the number of days you wish the unit to" +
                         " delay");
                INPUT(real);
                ASK SELF TO SetDelayUntil(real);
                EXIT;
        ELSIF (CHR = "N") OR (CHR = "n")
                EXIT;
        END IF
END LOOP;

END METHOD;
```

```
{-----------------------------------------------------------------------}
TELL METHOD Consume;
{-----------------------------------------------------------------------}

{EVERY 24.0 TIME UNITS, A UNIT CHECKS ITS INVENTORY AND REDUCES EACH COMMODITY
ONHAND BY THE CURRNET CONSUMPTION RATE}

VAR

CurrentItem : CommodityObj;
i, numItems : INTEGER;

{Go Thru Inventory, item by item, decrementing by daily usage rate (or rv)
ordering if required}

BEGIN

ASK SELF TO DumpFields;

LOOP
        WAIT DURATION 24.0

ASK SELF TO DumpFields;

                numItems := ASK Inventory numberIn;

                ASK SELF TO SetInPlace("TRUE");

                FOR i := 1 TO numItems
                        CurrentItem := ASK Inventory TO Remove;
                        ASK Inventory TO Add(CurrentItem);
                        IF (NOT CurrentItem.InPlace)AND(CurrentItem.Deployment)
                                ASK SELF TO SetInPlace("FALSE");
                        END IF;
                END FOR;

                FOR i := 1 TO numItems
                        CurrentItem := ASK Inventory TO Remove;
                        ASK Inventory TO Add(CurrentItem);
                        IF InPlace
                        CASE (ASK SELF CombatIntensity)
                                WHEN Low :
                                        ASK CurrentItem TO SubtractOnHand
                                 (MINOF(CurrentItem.OnHand,CurrentItem.LowRate));
                                WHEN Med  :
                                        ASK CurrentItem TO SubtractOnHand
                                 (MINOF(CurrentItem.OnHand,CurrentItem.MedRate));
                                WHEN High :
                                        ASK CurrentItem TC SubtractOnHand
                                (MINOF(CurrentItem.OnHand,CurrentItem.HighRate));
                                WHEN None :
                                        ASK CurrentItem TO SubtractOnHand
                                (MINOF(CurrentItem.OnHand,CurrentItem.NoneRate));
                                OTHERWISE
                                        OUTPUT("INVALID COMBAT INTENSITY");
                                        HALT;
                        END CASE;

                                                    {Set Higher Priority if
```

```
                                                        necessary}
                        IF (CurrentItem.OnHand <= CurrentItem.EmerOrderAt)
                                ASK CurrentItem TO
                                         SetPriority(CurrentItem.EmerPriority);
                        ELSE
                                ASK CurrentItem TO
                                        SetPriority(CurrentItem.NormalPriority);
                        END IF;

                        END IF;
                        IF CurrentItem.OnHand >= ActiveAt * CurrentItem.StockTo
                                ASK CurrentItem TO SetInPlace("TRUE");
                        END IF;
                        IF CurrentItem.OnHand + CurrentItem.OnOrder <
                                                        CurrentItem.OrderAt
                                ASK SELF TO OrderStuff(CurrentItem);
                        END IF;
                END FOR;
{WriteLine(Name + "Comsuming at time:");}
        ON INTERRUPT;
        END WAIT;
END LOOP;
END METHOD;

{--------------------------------------------------------------------------}
TELL METHOD DelayActivation;
{--------------------------------------------------------------------------}

{METHOD ALLOW UNITS TO DELAY FOR A SPECIFIC TIME PERIOD BEFORE ACTIVATING}


BEGIN
IF DelayUntil < 0.0
        DelayUntil := 0.0;
END IF;
WAIT DURATION (DelayUntil * 24.0);

        TELL SELF TO Consume;

ON INTERRUPT
END WAIT;

END METHOD;

{--------------------------------------------------------------------------}
     ASK METHOD Display;
{--------------------------------------------------------------------------}
CONST

format = "          *********** ********** ********** ********** **********";

format2 = "***. **************** ********** ********** ********** ********** *****";
tit. ; = "======================= ***************> at time ******<  ============

VAR

j, i, numItems : INTEGER;
Transporter : TransporterObj;
Commodity : CommodityObj;
string, answer : STRING;
```

```
CHR : CHAR;

BEGIN

ClearScreen;
j := 0;
SOUTPUT(" ",j);
string := SPRINT(Name, TRUNC(SimTime)) WITH title;
SOUTPUT(string,j);
SOUTPUT(" ", j);

OutputPosition(Position,j);
SOUTPUT(" ",j);

OUTPUT("Unit Class:  ", Class, "     Combat Intensity:  ", CombatIntensity);
j := j + 1;
IF InPlace
        SOUTPUT("In Place", j);
ELSE
        SOUTPUT("Closing", j);
END IF;

SOUTPUT(" ", j);

IF Linked
        SOUTPUT("Origin:  "+ Origin.Name, j);
END IF;

IF HasAirPort
        SOUTPUT(" ", j);
        SOUTPUT("Airport:                  Max Capacity: "
                 + INTTOSTR(AirPort.MaxCapacity) + "     Max Vehicle Size: "
                 + REALTOSTR(AirPort.MaxSize),j);
        OUTPUT("              Arrivals:   ", AirPort.ArrivalsQ.numberIn);
        OUTPU`              Ramp:       ", AirPort.BerthsQ.numberIn);
        OUTPUT("              Parked:     ", AirPort.ParkedQ.numberIn);
        j := j + 3;

END IF;

IF HasSeaPort
        SOUTPUT(" ",j);
        SOUTPUT("Seaport:                  Max Capacity: "
                 + INTTOSTR(SeaPort.MaxCapacity)+ "     Max Vehicle Size: "
                 + REALTOSTR(SeaPort.MaxSize),j);
        OUTPUT("              Arrivals:   ", SeaPort.ArrivalsQ.numberIn);
        OUTPUT("              Berths:     ", SeaPort.BerthsQ.numberIn);
        OUTPUT("              Anchorage:  ", SeaPort.ParkedQ.numberIn);
        j := j + 3;

END IF;


IF HasRail
        SOUTPUT(" ",j);
        SOUTPUT("Railyard:                 Max Capacity: "+
                INTTOSTR(RailYard.MaxCapacity)+ "     Max Vehicle Size: "+
                                              REALTOSTR(RailYard.MaxSize),j);
        OUTPUT("              Arrivals: ", RailYard.ArrivalsQ.numberIn);
        OUTPUT("              Station:  ", RailYard.BerthsQ.numberIn);
        OUTPUT("          Yard:      ", RailYard.ParkedQ.numberIn);
        j := j + 3;
END IF;

IF HasTruckStop
        SOUTPUT(" ",j);
        SOUTPUT("Truck Stop:            Max Capacity: "
                 + INTTOSTR(TruckStop.MaxCapacity)+ "      Max Vehicle Size: "
                 + REALTOSTR(TruckStop.MaxSize), j);
        OUTPUT("          Arrivals:  ", TruckStop.ArrivalsQ.numberIn);
        OUTPUT("          Terminal:  ", TruckStop.BerthsQ.numberIn);
        OUTPUT("          Parked:    ", TruckStop.ParkedQ.numberIn);
        j := j + 3;
END IF;

SOUTPUT(" ", j);

numItems := ASK Inventory numberIn;
SOUTPUT("Items in Inventory:  "+ INTTOSTR(numItems), j);
SOUTPUT("                       ON HAND  STOCK TO  ORDER AT  ON ORDER DEPLOY",
         j);

FOR i := 1 TO numItems
        Commodity := ASK Inventory TO Remove;
        ASK Inventory TO Add(Commodity);
string := SPRINT(i, Commodity.Name, TRUNC(Commodity.OnHand),
         TRUNC(Commodity.StockTo), TRUNC(Commodity.OrderAt),
         TRUNC(Commodity.OnOrder), Commodity.Deployment) WITH format2;
SOUTPUT(string, j);
END FOR;

SOUTPUT("==================================================================
SOUTPUT(" ", j);

END METHOD;

{----------------------------------------------------------------------}
     ASK METHOD Modify(INOUT builder : BuilderObj);
{----------------------------------------------------------------------}

{ALLOWS INTERACTIVE MODIFICATION OF A UNITS FIELDS}

CONST

format =
"*************** ******.*** ******.*** ******.*** ******.*** ********* *****";

VAR

base : BaseObj;
string, answer, answer2: STRING;
TransporterQ, BigTransporterQ : TransporterQObj;
Transporter, NewTransporter : TransporterObj;
NewPort, port : PortObj;
commodity : CommodityObj;
integer : INTEGER;
real : REAL;
j, i, numItems : INTEGER;
CHR, CHR2 : CHAR;
position : PositionRecType;
```

```
BEGIN

        LOOP
                ASK SELF TO Display;
                OUTPUT("       MODIFY?  (P)orts, (I)nventory, (S)tatus, (R)eturn"
                CHR := ReadKey();
                IF (CHR = "P") OR (CHR = "p")
                        LOOP

                        ClearScreen;
                        OUTPUT(" ");
                        OUTPUT("       ",SELF.Name, " has the following ports:");
                        OUTPUT(" ");
                        OUTPUT("            Airport:      ", SELF.HasAirPort);
                        OUTPUT("            Seaport:      ", SELF.HasSeaPort);
                        OUTPUT("            Railyard:     ", SELF.HasRail);
                        OUTPUT("            Truck Stop:   ", SELF.HasTruckStop);
                        OUTPUT(" ");
                        OUTPUT("       MODIFY?   (A)irport, (S)eaport, Rail(y)ard,
                        CHR2 := ReadKey();
                        IF (CHR2 = "A") OR (CHR2 = "a")
                                port := ASK SELF AirPort;
                        ELSIF (CHR2 = "S") OR (CHR2 = "s")
                                port := ASK SELF SeaPort;
                        ELSIF (CHR2 = "Y") OR (CHR2 = "y")
                                port := ASK SELF RailYard;
                        ELSIF (CHR2 = "T") OR (CHR2 = "t")
                                port := ASK SELF TruckStop;
                        ELSIF (CHR2 = "R") OR (CHR2 = "r");
                                EXIT;
                        END IF;
                        OUTPUT("MODIFY? (E)xistence, Max (C)apacity, Max Vehicle
                        OUTPUT("        (T)ransporters" );
                        CHR := ReadKey();
                        IF (CHR = "E") OR (CHR = "e")
                                IF (CHR2 = "A") OR (CHR2 = "a")
                                        IF SELF.HasAirPort
                                        ASK SELF TO SetHasAirPort("FALSE");
                                        ELSE
                                        ASK SELF TO SetHasAirPort("TRUE");
                                        END IF;
                                ELSIF (CHR2 = "S") OR (CHR2 = "s")
                                        IF SELF.HasSeaPort
                                        ASK SELF TO SetHasSeaPort("FALSE");
                                        ELSE
                                        ASK SELF TO SetHasSeaPort("TRUE");
                                        END IF;
                                ELSIF (CHR2 = "Y") OR (CHR2 = "y")
                                        IF SELF.HasRail
                                        ASK SELF TO SetHasRail("FALSE");
                                        ELSE
                                        ASK SELF TO SetHasRail("TRUE");
                                        END IF;
                                ELSIF (CHR2 = "T") OR (CHR2 = "t")
                                        IF SELF.HasTruckStop
                                        ASK SELF TO SetHasTruckStop("FALSE");
                                        ELSE
                                        ASK SELF TO SetHasTruckStop("TRUE");
```

```
                    END IF;
                END IF;


            ELSIF (CHR = "C") OR (CHR = "c")
                OUTPUT("      Enter New Max Capacity and hit <ENT
                INPUT(integer);
                ASK port TO SetMaxCapacity(integer);
            ELSIF (CHR = "S") OR (CHR = "s")
                OUTPUT("      Enter New Max Vehicle Size (Real)."
                OUTPUT("      Units:  Air    - Square foot area");
                OUTPUT("              Ship   - Overall length (fee
                OUTPUT("              Rail   - Length in cars.");
                OUTPUT("              Truck  - Vehicles in Convoy.
                INPUT(real);
                ASK port TO SetMaxSize(real);

            ELSIF (CHR = "N") OR (CHR = "n")
            LOOP
                ClearScreen;
                OUTPUT(" ");
                OUTPUT("      Transportation network includes the
                OUTPUT(" ");
                numItems := ASK port.Network numberIn;
                IF numItems <> 0
                FOR i := 1 TO numItems
                        base := ASK port.Network TO Remove;
                        ASK port.Network TO Add(base);
                        OUTPUT(base.Name);
                END FOR;
                ELSE
                        OUTPUT("      NONE");
                END IF;
                OUTPUT(" ");
                OUTPUT("      COMMAND:  (A)dd, (S)ubtract, (R)etu
                CHR2 := ReadKey();

                IF (CHR2 = "A") OR (CHR2 = "a")
                        OUTPUT("Base or Unit Name?");
                        INPUT(answer);
                        base := ASK builder.BaseQ TO FindByName(
                        IF base <> NILOBJ
                                ASK port.Network TO Add(base);
                        END IF;
                ELSIF (CHR2 = "S") OR (CHR2 = "s")
                        OUTPUT("Base or Unit Name?");
                        INPUT(answer);
                        base := ASK builder.BaseQ TO FindByName(
                        IF base <> NILOBJ
                                ASK port.Network TO     RemoveTh
                        END IF;
                ELSIF (CHR2 = "R") OR (CHR2 = "r")
                        EXIT;
                END IF;
            END LOOP;
            ELSIF (CHR = "T") OR (CHR = "t")
                OUTPUT("      (A)dd or (D)elete");
                CHR := ReadKey();
                IF (CHR = "A") OR (CHR = "a")
                    ClearScreen;
```

```
                  j := 0;
                  IF Builder <> NILOBJ
                      TransporterQ := Builder.TransporterQ;
                      BigTransporterQ :=
                               Builder.BigTransporterQ;
                  ELSIF ScenarioEditor <> NILOBJ
                      TransporterQ :=
                               ScenarioEditor.TransporterQ;
                      BigTransporterQ :=
                               ScenarioEditor.BigTransporterQ;
                  END IF;
                  IF TransporterQ <> NILOBJ
                      ClearScreen;
                      j := 0;
                      ASK TransporterQ TO Display(j);
                      OUTPUT("     Input transporter name.");
                      INPUT(string);
                      Transporter := ASK TransporterQ TO
                               FindByName(string);
                      IF Transporter <> NILOBJ
                          OUTPUT("     How many?");
                          INPUT(integer);

                          FOR i := 1 TO integer
                              NewTransporter :=
                               CLONE(Transporter);
                              ASK Transporter TO
                      SetVehicleID(Transporter.VehicleID + 1);
                              ASK NewTransporter TO
                          SetVehicleID(Transporter.VehicleID);
                              ASK NewTransporter TO
                               SetLocation(SELF);
                              ASK NewTransporter TO
                               SetPosition(Position);
                              ASK NewTransporter TO
                               SetPort(SELF);
                              ASK BigTransporterQ TO
                               Add(NewTransporter);
                              CASE ASK NewTransporter Class
                              WHEN Aircraft :
                                      NewPort := AirPort;
                              WHEN Ship :
                                      NewPort := SeaPort;
                              WHEN Rail :
                                      NewPort := RailYard;
                              WHEN Truck :
                                      NewPort := TruckStop;
                              END CASE;
                              ASK NewPort.ParkedQ TO
                               Add(NewTransporter);
                              ASK TransporterManager TO
                  ReceiveAvailableTransporter(NewTransporter);
                          END FOR;
                      END IF;
                  END IF;
              ELSIF (CHR = "D") OR (CHR = "d")
                  ClearScreen;
                  j := 0;
                  ASK port.ParkedQ TO Display(j);
                  OUTPUT("     Input Name");
```

```
                                    INPUT(string);
                                    OUTPUT("     Input ID");
                                    INPUT(integer);
                                    numItems := ASK port.ParkedQ numberIn;
                                    FOR i := 1 TO numItems
                                        Transporter := ASK port.ParkedQ TO
                                                Remove;
                                        ASK port.ParkedQ TO Add(Transporter);
                                        IF Transporter.VehicleID = integer;
                                                ASK port.ParkedQ TO
                                                        RemoveThis(Transporter);
                                                OUTPUT(Transporter.Name,
                                                         Transporter.VehicleID,
                                                         " deleted");
                                                ASK Transporter TO CleanUp;
                                        END IF;
                                    END FOR;
                                END IF;
                        ELSIF (CHR = "R") OR (CHR = "r")
                                EXIT
                        END IF;

                        END LOOP;


                ELSIF (CHR = "I") OR (CHR = "i")
                    OUTPUT("     (A)dd Commodity, (E)dit Commodity?");
                    CHR := ReadKey();
                    IF (CHR = "A") OR (CHR = "a")
                        InputCommodities(builder);
                    ELSIF (CHR = "E") OR (CHR = "e")
                        OUTPUT("     Enter Commodity Name then hit <ENTER>");
                        INPUT(string);
                        commodity := ASK SELF.Inventory TO FindByName(string);
                        OUTPUT(" ");
                        IF commodity <> NILOBJ
                            LOOP
                                ClearScreen;
                                OUTPUT("");
OUTPUT("Name                  High     Medium        Low       None  Stock To"
         + " Deploy");
OUTPUT("==========================================================================
                                string := SPRINT(commodity.Name,
                                         commodity.HighRate, commodity.MedRate,
                                         commodity.LowRate, commodity.NoneRate,
                                         commodity.StockTo,
                                         commodity.Deployment) WITH format;
                                OUTPUT(string);
                                OUTPUT(" ");
                                OUTPUT("     MODIFY? (O)n Hand, (S)tocking" +
                                        " Objective, Order (P)oint," +
                                        " (C)onsumption");
                                OUTPUT("     (D)eployment, (R)eturn");
                                CHR := ReadKey();


                                IF (CHR = "O") OR (CHR = "o")
                                        OUTPUT("     Input new amount On Hand.")
                                        INPUT(real);
                                        ASK commodity TO SetOnHand(real);
                ELSIF (CHR = "S") OR (CHR = "s")
                        OUTPUT("     Input new Stocking Objectiv
                        INPUT(real);
                        ASK commodity TO SetStockTo(real);
                ELSIF (CHR = "P") OR (CHR = "p")
                        OUTPUT("     Input new Order Point.");
                        INPUT(real);
                        ASK commodity TO SetOrderAt(real);
                ELSIF (CHR = "C") OR (CHR = "c")
        OUTPUT("     (H)igh, (M)edium, (L)ow, (N)one");
                CHR := ReadKey();
        IF (CHR = "H") OR (CHR = "h")
                OUTPUT("     How much do you the subunit" +
                        " to consume per day in Heavy Combat?");
                INPUT(real);
                ASK commodity TO SetHighRate(real);
        ELSIF (CHR = "M") OR (CHR = "m")
                OUTPUT("     How much do you want the subunit" +
                        " to consume per day in Combat?");
                INPUT(real);
                ASK commodity TO SetMedRate(real);
        ELSIF (CHR = "L") OR (CHR = "l")
                OUTPUT("     How much do you want the subunit" +
                        " to consume per day in Light Combat?");
                INPUT(real);
                ASK commodity TO SetLowRate(real);
        ELSIF (CHR = "N") OR (CHR = "n")
                OUTPUT("     How much do you want the subunit" +
                        " to consume per day, No Combat?");
                INPUT(real);
                ASK commodity TO SetNoneRate(real);
        ELSIF (CHR = "R") OR (CHR = "r")
        END IF;
                ELSIF (CHR = "D") OR (CHR = "d")
                        LOOP
                        OUTPUT("     Do you want this commodity
                        INPUT(CHR);
                        IF (CHR = "Y") OR (CHR = "y")
                                ASK commodity TO
                                         SetDeployment("TRUE");
                                EXIT;
                        ELSIF (CHR = "N") OR (CHR = "n")
                                ASK commodity TO
                                         SetDeployment("FALSE");
                                EXIT;
                        END IF;
                        END LOOP;
                ELSIF (CHR = "R") OR (CHR = "r")
                        EXIT;

                END IF;
            END LOOP;
        END IF;
    END IF;
ELSIF (CHR = "S") OR (CHR = "s")
        LOOP

        OUTPUT("     MODIFY? Combat (I)ntensity, Closure (S)tatu
        CHR := ReadKey();
        IF (CHR = "I") OR (CHR = "i")
```

```
                                        OUTPUT("     (H)igh, (M)edium, (L)ow, (N)one");
                                        CHR := ReadKey();
                                        IF (CHR = "H") OR (CHR = "h")
                                                ASK SELF TO SetCombatIntensity("High");
                                        ELSIF (CHR = "M") OR (CHR = "m")
                                                ASK SELF TO SetCombatIntensity("Med");
                                        ELSIF (CHR = "L") OR (CHR = "l")
                                                ASK SELF TO SetCombatIntensity("Low");
                                        ELSIF (CHR = "N") OR (CHR = "n")
                                                ASK SELF TO SetCombatIntensity("None");
                                        END IF;


                                ELSIF (CHR = "S") OR (CHR = "s")
                                        OUTPUT("     (I)n Place or (C)losing?");
                                        CHR := ReadKey();
                                        IF (CHR = "I") OR (CHR = "i")
                                                ASK SELF TO SetInPlace("TRUE");
                                        ELSIF (CHR = "C") OR (CHR = "c")
                                                ASK SELF TO SetInPlace("FALSE");
                                        END IF;


                                ELSIF (CHR = "P") OR (CHR = "p")
                                        ClearScreen;
                                        j := 0;
                                        OutputPosition(Position, j);
                                        position := InputPosition();
                                        ASK SELF TO SetPosition(position);
                                        OutputPosition(Position, j);
                                ELSIF (CHR = "A") OR (CHR = "a")
                                        ASK SELF TO Activate;

                                ELSIF (CHR = "R") OR (CHR = "r")
                                        EXIT;
                                END IF;

                                END LOOP;

                        ELSIF (CHR = "R") OR (CHR = "r")
                                EXIT;
                        END IF;
                END LOOP;

        END METHOD;


{-----------------------------------------------------------------------}
        ASK METHOD OrderStuff(INOUT Item : CommodityObj);
{-----------------------------------------------------------------------}

{SENDS ORDER TO LOGISTICS MANAGER WHEN SHORTFALL IN INVENTORY IS DISCOVERED}

        VAR
            ItemClass : CommodityClassType;

        BEGIN
```

```
{
OUTPUT("IN OrderStuff - ", Name);
}
         ASK LogisticsManager TO HandleUnitRequest(SELF, Item);

     END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD FillOrder(INOUT Shipment : ShipmentObj);
{---------------------------------------------------------------------------}
     BEGIN
     END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD BackOrderStuff(INOUT BackOrderShipment : ShipmentObj);
{---------------------------------------------------------------------------}
     BEGIN
     END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD FillBackOrders;
{---------------------------------------------------------------------------}
     BEGIN
     END METHOD;

{---------------------------------------------------------------------------}
     TELL METHOD CheckInventory;
{---------------------------------------------------------------------------}
     BEGIN
     END METHOD;

{---------------------------------------------------------------------------}
     ASK METHOD ObjInit;
{---------------------------------------------------------------------------}
BEGIN

INHERITED ObjInit;
{TELL SELF TO Consume;}

END METHOD;


END OBJECT;

END MODULE.
```

```
DEFINITION MODULE WriteLine;

PROCEDURE WriteLine(IN String : STRING);

END MODULE.
```

```
IMPLEMENTATION MODULE WriteLine;

FROM IOMod IMPORT FileUseType(Output);
FROM IOMod IMPORT StreamObj;
FROM UtilMod IMPORT DateTime;

VAR
DT : STRING;
TraceStream : StreamObj;

PROCEDURE WriteLine(IN String : STRING);

BEGIN
IF (TraceStream = NILOBJ)
NEW(TraceStream);
ASK TraceStream TO Open("sim.out", Output);
DateTime(DT);
ASK TraceStream TO WriteString(DT);
ASK TraceStream TO WriteLn;
ASK TraceStream TO WriteLn;
END IF;

ASK TraceStream TO WriteString(String);
ASK TraceStream TO WriteLn;

END PROCEDURE;


{OLD VERSION}
{
FROM Debug IMPORT TraceStream;
FROM IOMod IMPORT FileUseType(Output);

FROM IOMod IMPORT StreamObj;

VAR
file : StreamObj;

PROCEDURE WriteLine(IN String : STRING);

BEGIN
IF (TraceStream = NILOBJ)
NEW(TraceStream);
ASK TraceStream TO Open("sim.out", Output);
ASK TraceStream TO TraceOff;
END IF;

ASK TraceStream TO WriteString(String);
ASK TraceStream TO WriteLn;

END PROCEDURE;
}


PROCEDURE WriteLineClose;

BEGIN
ASK TraceStream TO Close;
END PROCEDURE;
END MODULE.
```

```
MAIN MODULE S3;

FROM DOSMenu IMPORT DOSMenu;

CONST

VAR

BEGIN

NEW(DOSMenu);
ASK DOSMenu TO DisplayMainMenu;

END MODULE;
```

LIST OF REFERENCES

1. Vlahos, Michael, "Wargaming. an Enforcer of Strategics Realism: 1919-1942," *Naval War College Review*, v. 39, p. 7, March-April 1986.

2. Perla, Peter, *The Art of Wargaming*, pp. 25-30, United States Naval Institute, 1990.

BIBLIOGRAPHY

1. Armed Forces Staff College, *AFSC PUB 1 The Joint Staff Officers Guide 1991*, April 1992.

2. Armed Forces Staff College, *AFSC PUB 7 ARMED FORCES STAFF COLLEGE REFERENCE BOOK*, April 1992.

3. CACI Products Company, *MODSIM II The Language of Object-Oriented Programming: Reference Manual*, 1992.

4. Center For Naval Analyses CRM-91-109, *Sealift in Operation Desert Shield/Desert Storm: 7 August 1990 to 17 February 1991*, by Ronald F. Rost, John F. Adams, and John J. Nelson, May 1991.

5. Headquarters, Department of the Army, *FM-101-10-1/2 STAFF OFFICERS' FIELD MANUAL ORGANIZATIONAL, TECHNICAL, AND LOGISTICAL DATA PLANING FACTORS*. 1987.

6. Military Traffic Management Command Transportation Engineering Agency, *LOGISTICS HANDBOOK FOR STRATEGIC MOBILITY PLANNING*, August 1989.

7. Military Traffic Management Command Transportation Engineering Agency, *DEPLOYMENT PLANNING GUIDE*, August 1991.

8. Perla, Peter, *The Art of Wargaming*, United States Naval Institute, 1990.

9. U.S. Army Command and General Staff College, *PLANNING FACTORS*, 1 June 1985.

10. Vlahos, Michael, "Wargaming. an Enforcer of Strategics Realism: 1919-1942," *Naval War College Review*, v. 39, March-April 1986.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Deputy Chief of Naval Operations, N402D ATTN: CDR Craig Turley, USN Department of the Navy Washington, D.C. 20350-2000	2
4.	Defense Logistics Studies Information Exchange U.S. Army Logistics Management Center Fort Lee, Virginia 23801	2
5.	President, Naval War College ATTN: WGD 334L 686 Cushing Road Newport, Rhode Island 02841-1207	2
6.	PROF David A. Schrady, Code OR/So Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	1
7.	PROF Thomas E. Halwachs, Code OR/Ha Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	1
8.	LT John A. Long c/o Mrs. Hester J. Long 242 W. Lemon Street Lancaster, Pennsylvania 17603	2

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Deputy Chief of Naval Operations, N402D ATTN: CDR Craig Turley, USN Department of the Navy Washington, D.C. 20350-2000	2
4.	Defense Logistics Studies Information Exchange U.S. Army Logistics Management Center Fort Lee, Virginia 23801	2
5.	President, Naval War College ATTN: WGD 334L 686 Cushing Road Newport, Rhode Island 02841-1207	2
6.	PROF David A. Schrady, Code OR/So Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	1
7.	PROF Thomas E. Halwachs, Code OR/Ha Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	1
8.	LT John A. Long c/o Mrs. Hester J. Long 242 W. Lemon Street Lancaster, Pennsylvania 17603	2